AFFTC-TIM-07-04

A
F
F
T
C

A LIMITED EVALUATION OF FULL SCALE CONTROL SURFACE DEFLECTION DRAG (HAVE FUN)

R. BRENT REINHARDT, Capt, USAF
Project Manager/Project Pilot

SEAN A. CELI, Maj, USAF
Flight Test Navigator

JEFFREY T. GERAGHTY, Maj, USAF
Project Pilot

JAMES W. STAHL, Maj, USAF
Project Pilot

VICTOR J. GLOVER, LT, USN
Project Pilot

GEOFFREY G. BOWMAN, Capt, USAF
Flight Test Engineer

JUNE 2007

FINAL TECHNICAL INFORMATION MEMORANDUM

AIR FORCE FLIGHT TEST CENTER
EDWARDS AIR FORCE BASE, CALIFORNIA
AIR FORCE MATERIAL COMMAND
UNITED STATES AIR FORCE

This Technical Information Memorandum (AFFTC-TIM-07-04, Have FUN) was prepared and submitted under Job Order Number M07C0200 by the Have FUN test team.

Prepared by:

[illegible]

R. BRENT REINHARDT
Captain, USAF
Project Manager/Project Test Pilot

[illegible]

SEAN A. CELI
Major, USAF
Project Flight Test Navigator

[illegible]

JEFFREY T. GERAGHTY
Major, USAF
Project Test Pilot

[illegible]

JAMES W. STAHL
Major, USAF
Project Test Pilot

[illegible]

VICTOR J. GLOVER
Lieutenant, USN
Project Test Pilot

[illegible]

GEOFFREY G. BOWMAN
Captain, USAF
Project Flight Test Engineer

Reviewed by:

[illegible]

CHAD E. RYTHER
Major, USAF
Staff Advisor

[illegible]

BRIAN A. KISH
Lieutenant Colonel, USAF
Test Management Branch Chief, USAF Test Pilot School

[illegible]

JOHN L. MINOR, YD3, DAF
Technical Director, USAF Test Pilot School

Approved by:

[illegible]

TERRY M. LUALLEN, Colonel, USAF
Commandant, USAF Test Pilot School

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE 9 June 2007	**2. REPORT TYPE** Final Technical Information Memorandum	**3. DATES COVERED** *(From – To)* 6 – 23 Mar 2007
4. TITLE AND SUBTITLE **A Limited Evaluation of Full Scale Control Surface Deflection Drag (Have FUN)**		**5a. CONTRACT NUMBER**
6. AUTHOR(S) Reinhardt, Brent, Captain, USAF; Celi, Sean, Major, USAF; Geraghty, Jeffrey, Major, USAF; Stahl, James, Major, USAF; Glover, Victor, Lieutenant, USN; Bowman, Geoffrey, Captain, USAF		**5b. GRANT NUMBER**
		5c. PROGRAM ELEMENT NUMBER
		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Flight Test Center 412th Test Wing USAF Test Pilot School 220 South Wolfe Ave Edwards AFB CA 93524-6485		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFFTC-TIM-07- 04
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) USAF Test Pilot School Building 1220 220 South Wolfe Ave Edwards AFB CA 93524-6485		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		
13. SUPPLEMENTARY NOTES CA: Air Force Flight Test Center Edwards AFB CA CC: 012100		

14. ABSTRACT

The Have FUN (FUll Scale Numbers) Test Management Project was conducted at the request of the USAF TPS as an investigation into the drag caused by control surface deflection during dynamic soaring techniques. Forty-three test sorties were performed from 6-23 March 2007 under Job Order Number (JON) M07C0200. The six member Have FUN test team from TPS Class 06B performed the testing at the North Base facilities of Edwards AFB. Flight testing gathered sailplane decelerations during control surface deflections and energy height data during the performance of Dynamic Soaring maneuvers and straight glides. Coefficients of drag for each control surface were determined from minimum sink to maneuvering airspeed of the L-23 Super Blanik.

15. SUBJECT TERMS

Have FUN, Dynamic Soaring, L-23 Super Blanik Aircraft, Flight Testing, Control Surface Drag, Atmospheric Energy Extraction, Sailplane Flight Data, Sailplanes, Energy Height Data, Soaring Maneuvers, Sailplane Accelerations, Control Surface Deflections, Deflection Drag, Drag Coefficients

16. SECURITY CLASSIFICATION OF: Unclassified			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Chad E. Ryther, Major, USAF
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAME AS REPORT	74	**19b. TELEPHONE NUMBER** *(include area code)* 661-277-8308

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

This page intentionally left blank.

PREFACE

The Have FUN Test Management Project team would like to recognize Mr. Bill Gray (USAF TPS) and Mr. James Murray (NASA Dryden) for their contribution to the successful completion of this project. Their efforts were instrumental to the completion of the Have FUN project.

This page intentionally left blank.

EXECUTIVE SUMMARY

This technical information memorandum presents the results of the Have FUN (FUll scale Numbers) Test Management Project. During this project the test team used a modified L-23 Super Blanik to perform forty-three test sorties from 6 to 23 March 2007. The L-23 modifications included the installation of a data acquisition system (DAS) and two tablet personal computers (PCs). The modifications did not significantly affect the flying qualities of the aircraft.

The project aimed to determine the drag contribution of fully deflected control surfaces during dynamic soaring maneuvers, to compare energy height losses during the performance of dynamic soaring maneuvers and straight glides, and to determine the effects of yaw, yaw rate, roll rate, and angle of attack (AOA) rate on drag. The first objective was met for the ailerons and the rudder, but not for the elevator. The second objective was met. The third objective was not met due to DAS limitations.

The test team used body axis decelerations resulting from full-scale deflections of each surface to determine drag coefficients for the rudder and the aileron. The drag coefficients were small but significant. Furthermore, a comparison of straight glides, a dynamic soaring maneuver, and a porpoise glide maneuver showed the smallest energy loss from the straight glide and the largest energy loss from the dynamic soaring maneuver. Finally, a synthesis of the first two objectives showed that the drag coefficients determined in objective one could predict approximately two-thirds of the energy height loss measured in objective two.

Although the drag contribution of control surface deflection was significant, it could not account for all of the energy lost during the execution of the dynamic soaring maneuver. Additional energy losses may be explained by the effects of yaw, yaw rate, roll rate, and AOA rate on drag, but could not be determined during this test. Because the dynamic soaring maneuver was nearly impossible to fly without inducing any sideslip, the most significant of these additional contributors was probably yaw. Future tests should use an improved DAS in an attempt to discover the effects of sideslip during the dynamic soaring maneuver.

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Background

Inexorably, the history of manned flight lifted off from ornithology, the study of birds. Otto Lilienthal, the seminal ornithologist-aviator, wrote of the practical application of his studies when he stated, "One can get a proper insight into the practice of flying only by actual flying experiments." (reference 1) Experimental test pilots have transcribed Mr. Lilienthal's prescient remarks to the 21st Century as they look once again toward experimental application of bird models in an attempt to gain a greater understanding of soaring techniques.

For example, the albatross, a bird native to the South Pacific, uses wind gradients close to the surface of the ocean to sustain flight without flapping its wings, a technique called "dynamic soaring". Similarly, remote control (RC) glider operators have been able to sustain un-powered flight and extract energy from wind gradients over a ridge. The dynamic soaring concept has provided the basis of multiple Test Pilot School (TPS) glider projects in the recent past, including project SENIOR IDS, a 2004 project that determined the lift and drag polars of the L-23 (reference 2), project SENIOR ShWOOPIN, a 2005 investigation into the utility of dynamic soaring (reference 3), and project ThUMP, a 2006 comparison of classical porpoising techniques to dynamic soaring (reference 4).

To further the ongoing investigation into dynamic soaring using the L-23, the USAF TPS requested an investigation into control surface deflection drag. The members of HAVE FUN from TPS Class 06B performed the tests at the Edwards AFB North Base facilities within airspace up to 10,500 feet pressure altitude. Additional personnel, including TPS staff and Skylark North (Tehachapi, CA) employees, complemented the HAVE FUN test team through their role as tow pilots and glider instructor pilots. All testing was accomplished under the Job Order Number (JON) M07C0200.

The members of HAVE FUN tested in the spirit of Otto Lilienthal, who concluded "The manner in which we have to meet the irregularities of the wind, when soaring in the air, can only be learnt by being in the air itself." (reference 1)

Program Chronology

Forty-three test sorties were performed in the L-23 between 6 and 23 March 2007. A total of 32.3 hours of flight test were accomplished.

Test Item Description

The L-23 Super Blanik (figure 1) was designed and manufactured by LET Aeronautics Works in the Czech Republic. The Super Blanik sailplane was a cantilever, high-wing, two-seat glider of all-metal structure (reference 4). The rudder, elevator, and ailerons were fabric covered. The T-tail was fitted with a conventional elevator and pitch trim tab for pitch control. The retractable main landing gear was fixed in the down position for the entire test and was a permanent modification. The L-23 lift to drag ratio (L/D), as reported by project SENIOR IDS (reference 2) was 24:1 at approximately 48 KIAS (dual aircrew) with the air brake retracted and the landing gear extended. The conventional three-axis flight control system was non-powered and fully reversible. Both cockpits were equipped with a center-mounted control stick and rudder pedals that actuated control surfaces using a combination of push rods and cables. The over- and under-wing air brakes were controlled by levers from either cockpit. The never exceed airspeed was 124 KIAS. Load factor limits were -2.6 to +5.3 g at full gross weight (1124 lbs with two occupants). For more information, see the flight manual (reference 5).

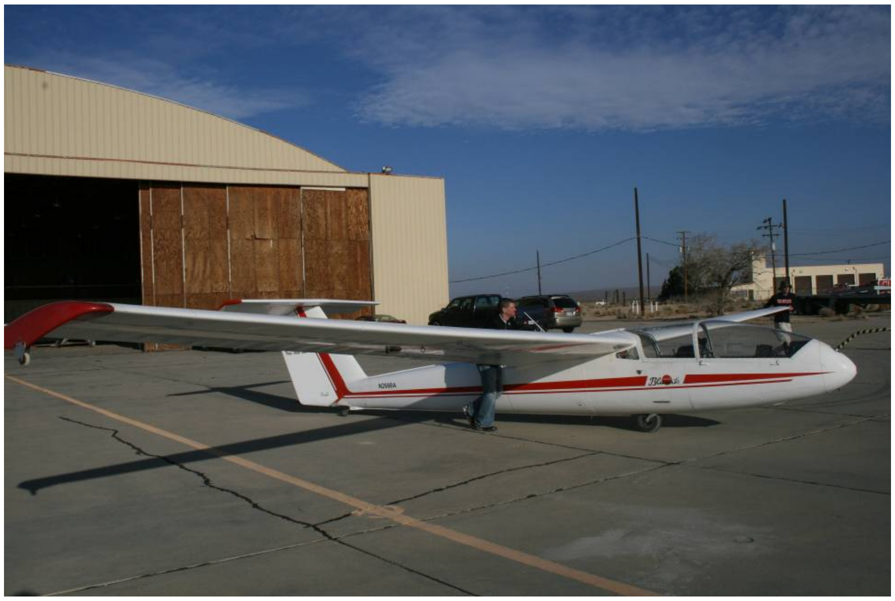


Figure 1: L-23 Super Blanik Test Aircraft

The aircraft was modified with a data acquisition system (DAS) consisting of a five-hole Pitot-static probe and an inertial measurement unit (IMU). Two tablet PCs displaying real time attitude, load factor, flight altitude, and specific energy height (E_S) information were added during previous projects. The rear cockpit contained a digital readout of energy height from the total energy variometer probe (figure 2).

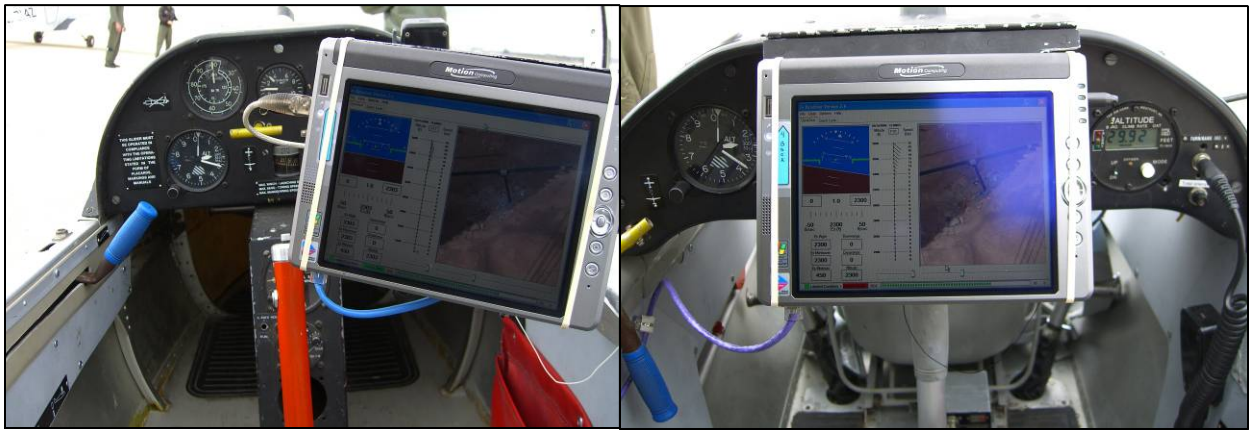


Figure 2: Front (left) and Rear (right) Cockpit Displays Panels

The total weight of modification equipment was 25 pounds, which allowed for a maximum combined weight of 393 pounds for crewmembers. The DAS was completely independent of the production Pitot-static system. The boom mounted five-hole Pitot-static probe had a hemispherical tip and measured total and differential pressure, provided airspeed, altitude, angle of attack (AOA), and angle of sideslip (AOSS) signals. The software on the tablet PC also provided the capability to playback recorded data post-flight. The IMU was installed in the baggage compartment behind the rear cockpit. The unit was a battery-powered GS-111m produced by Athena Technologies, Inc, Warrenton, VA. It incorporated the sensor suite necessary to provide a full attitude, navigation, and air data solution for use in vehicle flight-state measurement.

The GS-111m was equipped with accelerometers, angular rate sensors, and magnetometers in all three axes, an internal GPS receiver, and air data sensors. A real-time, multi-state Kalman filter was used to integrate the different sensors. The aircraft also had a VHF radio and was equipped with a Mode 3/C transponder system. See Appendix A: Instrumentation and Displays for more detailed information. For the purposes of this test, the L-23 was considered production representative.

Test Objectives

The overall test objectives for this project are listed below:

1. Determine the drag contribution of control surface deflection.

2. Compare specific energy loss during maneuvering flight to specific energy loss during wings-level glide.

3. Determine the drag contribution of sideslip, yaw rate, roll rate, and angle-of-attack rate.

Test objective 1 was met for the rudder and ailerons, but elevator contribution to drag was not determined. Objective 2 was met. Test objective 3 was not met.

Limitations

The elevator contribution to drag was not determined due to the inability to decouple multiple effects of elevator inputs. Specifically, the elevator inputs caused compounded increases and decreases in drag that could not be decoupled from the AOA rates caused by the same elevator inputs.

Additionally, the incremental deceleration due to sideslip, yaw rate, roll rate, and angle-of-attack rate was too small to extract from the noise present in the DAS using the data reduction techniques planned for this test program.

This page intentionally left blank.

TEST AND EVALUATION

Test Procedures

The test procedures consisted of the following basic steps. First, the members of the test team gathered at the Test Pilot School to conduct a pre-test briefing. The pre-test briefing included a review of the test matrix status, test objectives and procedures, aircraft and instrumentation status, previous abnormal test events, a safety review, and a weather and NOTAM review. The team proceeded to the North Base operating area, Edwards AFB. Tests were conducted at North Base as described in the test execution section, below. Flying operations continued during the entire available flying window.

Upon arrival at North Base, the test team performed an aircraft pre-flight inspection and an instrumentation check before pushing the L-23 and the tow plane out of the hangar. The test team contacted Edwards AFB ground control via radio and the North Base Glider Operations Area (figure 3) was activated. Before the first launch of the day, the Edwards control tower (Tower) was notified that the tow plane and gliders were starting operations.

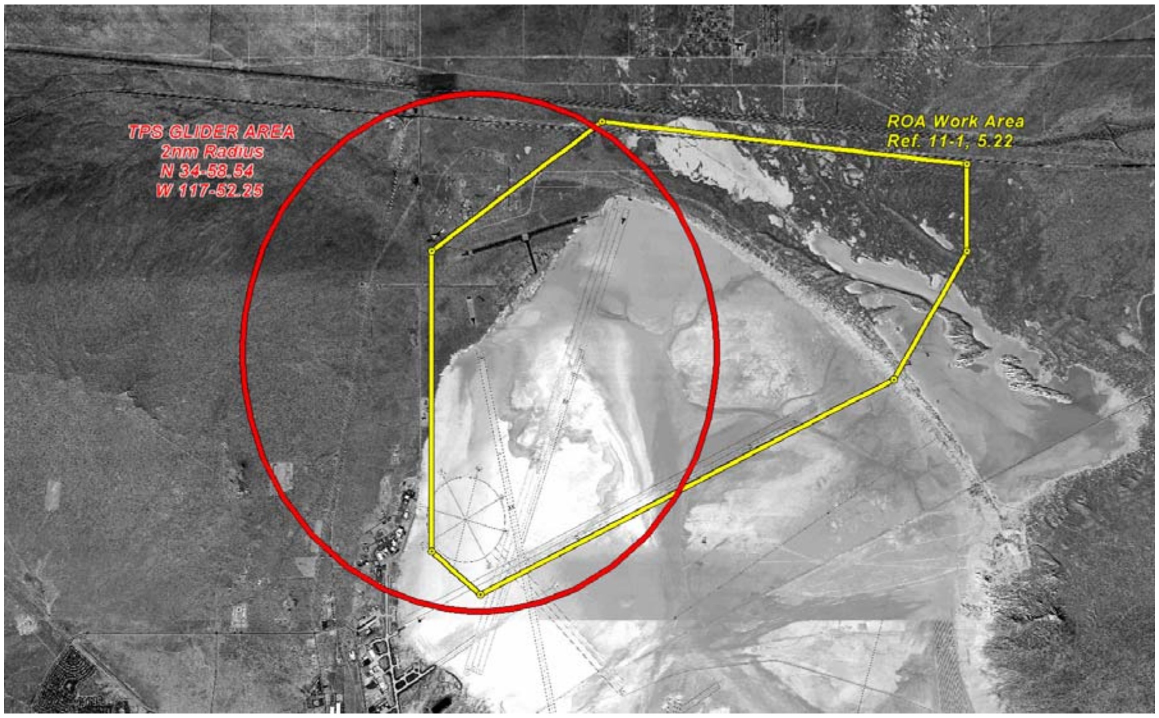


Figure 3: Glider Operating Area at North Base

The test team accomplished a modified glider aircrew checklist (appendix F) for each L-23 flight. When preflight checks were complete, the glider aircrew signaled for launch, and the glider was subsequently towed. The glider pilot released at the briefed altitude.

Once the last sortie of the day landed, tower was notified that all operations were complete. The test team returned the glider and tow plane to the hangar, and completed a daily flight log.

Have FUN Flight Test Techniques

To gather data for control surface deflection drag and drag due to sideslip, roll rate, yaw rate, and angle-of-attack rate, the test team performed three separate maneuvers beginning at a set airspeed in a wings-level attitude. First, the test pilot performed small-angle rolls at different speeds using one-third, two-thirds, and full deflection step inputs. The bank-to-bank rolls were accomplished using fixed rudder and elevator. Second, the pilot fixed the aileron and elevator position and input rudder step deflections. The rudder step deflections at speeds below max lift to drag ratio (L/D max) used one-third, two-thirds, and full deflection, whereas speeds at or above L/D max used one-fourth and one-half rudder deflection. Rudder inputs were held for approximately two seconds and terminated prior to stall indications. Third, the test pilot fixed the aileron and rudder and performed elevator step deflections using one-half deflection in the forward stick direction (no less than 0g) and one-third, two-thirds, and full stick deflection in the aft direction. Step inputs were held for approximately two seconds and terminated prior to stall indications. Each of the three maneuvers was repeated at 80, 70, 60, 50, and 40 knots.

To gather data for objective 2, the test team performed three more separate maneuvers for comparison. The first maneuver was a dynamic soaring maneuver, described as follows. After stabilizing at maneuvering airspeed (81 KIAS), the pilot smoothly pulled the nose up to approximately 25 degrees while turning approximately 30 degrees heading. As airspeed decreased in the climb, the pilot reversed the turn and rolled the aircraft to approximately 60 degrees of bank arriving at the maneuver apex back on entry heading with the nose on the horizon. The pilot then allowed the aircraft to float over the top as the nose dropped below the horizon and maintained a minimum airspeed of 40 KIAS. The pilot let the nose drop to about a 25 degrees nose low attitude while turning approximately 30 degrees in the opposite direction of the initial turn. As altitude decreased, the pilot reversed the roll and pulled smoothly to roll out on the entry heading, altitude, and airspeed to finish the maneuver in 1g straight and level flight (See figure 4).

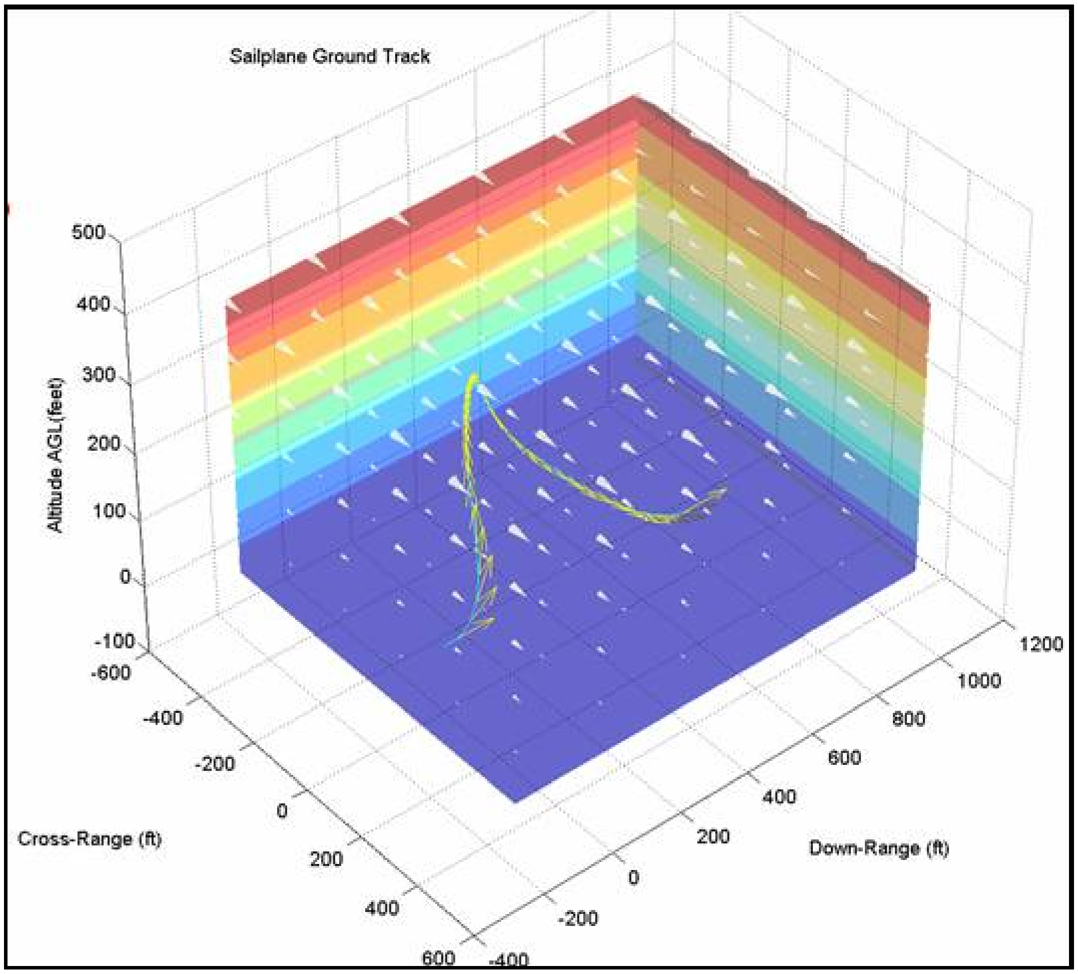


Figure 4: Dynamic Soaring Maneuver

The second maneuver was simply a straight glide at various airspeeds.

Third, a wings level ‘porpoise’ maneuver was performed by varying the airspeed with pitch alone in order to mimic airspeed changes noted during the dynamic soaring maneuver. In other words, because the airspeed varied during the dynamic soaring maneuver by slowing from 81 KIAS to 40 KIAS in 6 seconds, and then increasing to 81 KIAS in 8 seconds, the test team used the same timing to vary the airspeed during the porpoise maneuver, this time without any bank.

Results and Analysis

This section presents the results of the Have FUN test sorties and the analysis of the presented data.

Assumptions

The following assumptions were made in order to complete the test objectives.

- The L-23 Super Blanik test aircraft was assumed to have a rigid body.

- For the airspeeds used during testing (40 KIAS-80 KIAS) calibrated airspeed was equal to equivalent airspeed.

- Atmospheric conditions were assumed to have negligible impact on test results.

- Difference in aircraft gross weight from one aircrew to another was assumed to be negligible.

Sources of error in data reduction

- The angle of attack, angle of sideslip, outside air temperature, and indicated airspeed were noisy on the DAS data files.

- During steady state conditions (zero roll acceleration), the trim value of the ailerons varied significantly. For example, the DAS data would indicate five degrees left deflection, zero degrees right deflection, and no change in roll degrees. The source of this uncertainty is unknown at this time.

- Increased dynamic pressures during higher speed maneuvering prevented the control surfaces from reaching full deflection with full stick displacement.

- The rapid aileron and rudder control deflections excited a structural mode in the L-23 Super Blanik. This mode lasted throughout the data collection times. The change in drag was plotted and then visually averaged through the structural oscillations. This result proved the rigid body assumption was in error. Figure C1 is a sample of the plotted raw data with the structural mode present in the X and Y body-axis accelerations.

Error Minimization

- Pre-maneuver limits were set when evaluating DAS data to ensure a stable starting reference and ensure maneuver quality. Incidental rudder deflections prior to aileron inputs of less than plus or minus 2 degrees were considered good data. Likewise, incidental aileron deflections prior to rudder inputs of less than plus or minus 2 degrees were considered good. Furthermore, if the pitch or sideslip were more than 2 degrees prior to control input, the maneuver was not used for data reduction.

- Air speed control in the glider was controlled by elevator deflection alone. Except for the special case where the desired speed corresponded with zero elevator deflection, there was additional drag on the aircraft because of the elevator. This effect was mitigated by using the difference in drag caused by surface deflection instead of total drag.

- The angle of attack, angle of sideslip, outside air temperature, and indicated airspeed parameters were filtered using a Butterworth filter and zero-phase digital filtering to reduce noise effects (see appendix C for the MATLAB filtering algorithm).

Drag Contribution of Control Surface Deflection

Drag contributions from control surface deflections were determined for the aileron and rudder. No useful data could be gathered from the elevator deflections due to reasons listed in the limitations section above. Deflection magnitudes were performed in accordance with the test matrix that was developed using Design of Experiments (DOE) techniques (see appendix B). The instantaneous deceleration following control surface input was used to calculate drag. Data collected came from the first few tenths of a second after a control input before the rates associated with that control surface had a chance to build. The data reduction procedure is detailed in appendix C.

The increase in drag due to aileron deflection was calculated and plotted in figure 5 below. This plot showed the increase of the aileron specific coefficient of drag ($C_{Daileron}$) due to the composite aileron control surface deflection. Composite aileron deflection was the sum of the up and down deflection in degrees divided by two. The data were non-dimensionalized to a combined aileron surface area of 49.72 square feet. The curve fitted to the data was a forced zero intercept, quadratic model using all the data points (i.e., not removing outliers). The equation for the curve and the associated 95 percent confidence intervals were:

$$C_{D_{aileron}} = 5.0*10^{-5}(\pm 7.0*10^{-5})*adef^{2} + 0.002(\pm 0.0016)*adef$$

Where: *adef* = composite aileron deflection (deg)

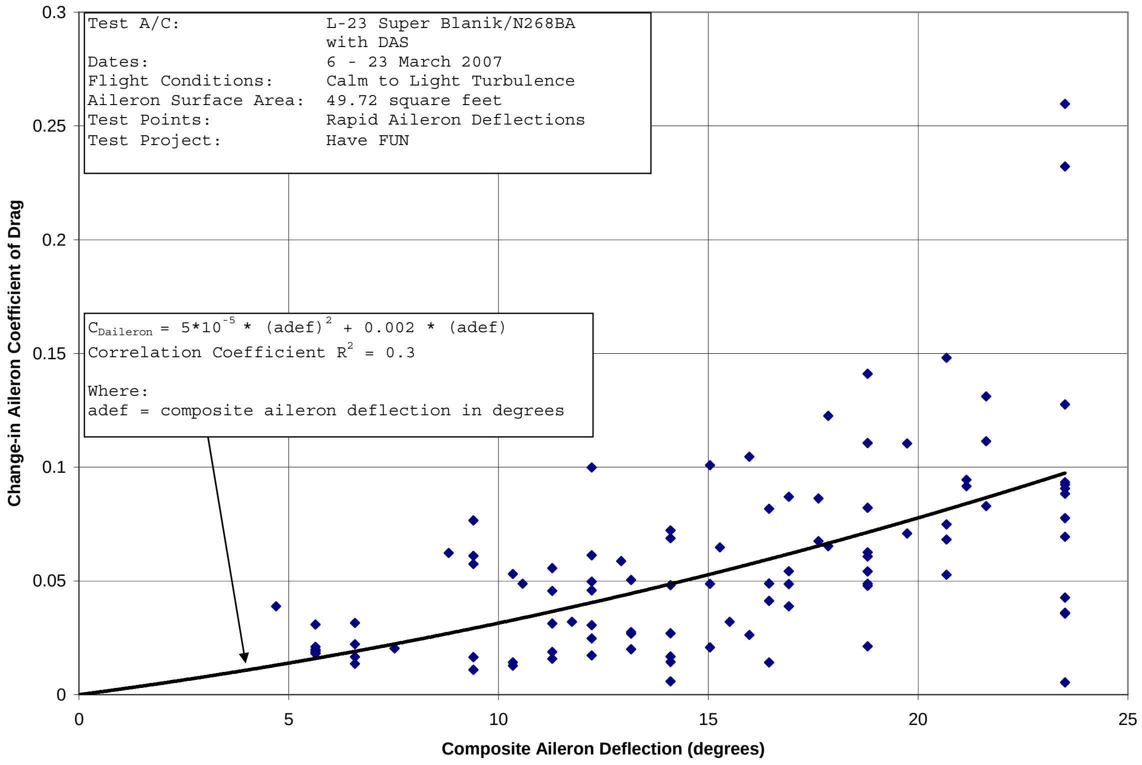


Figure 5: Aileron Deflection Induced Increase in $C_{Daileron}$

The increase in drag due to rudder deflection was calculated and plotted in figure 6 below. This plot showed the increase of $C_{Drudder}$ due to the rudder control surface deflection. The data were non-dimensionalized to a rudder surface area of 10.98 square feet. The curve fitted to the data was a forced zero intercept, quadratic model using all the data points (i.e., not removing outliers). The equation for the curve and the associated 95 percent confidence intervals were:

$$C_{Drudder} = 6.0 * 10^{-4}\ (\pm 1.9 * 10^{-4}) * rdef^2 - 0.0034\ (\pm\ 0.0047) * rdef$$

Where: rdef = rudder deflection (deg)

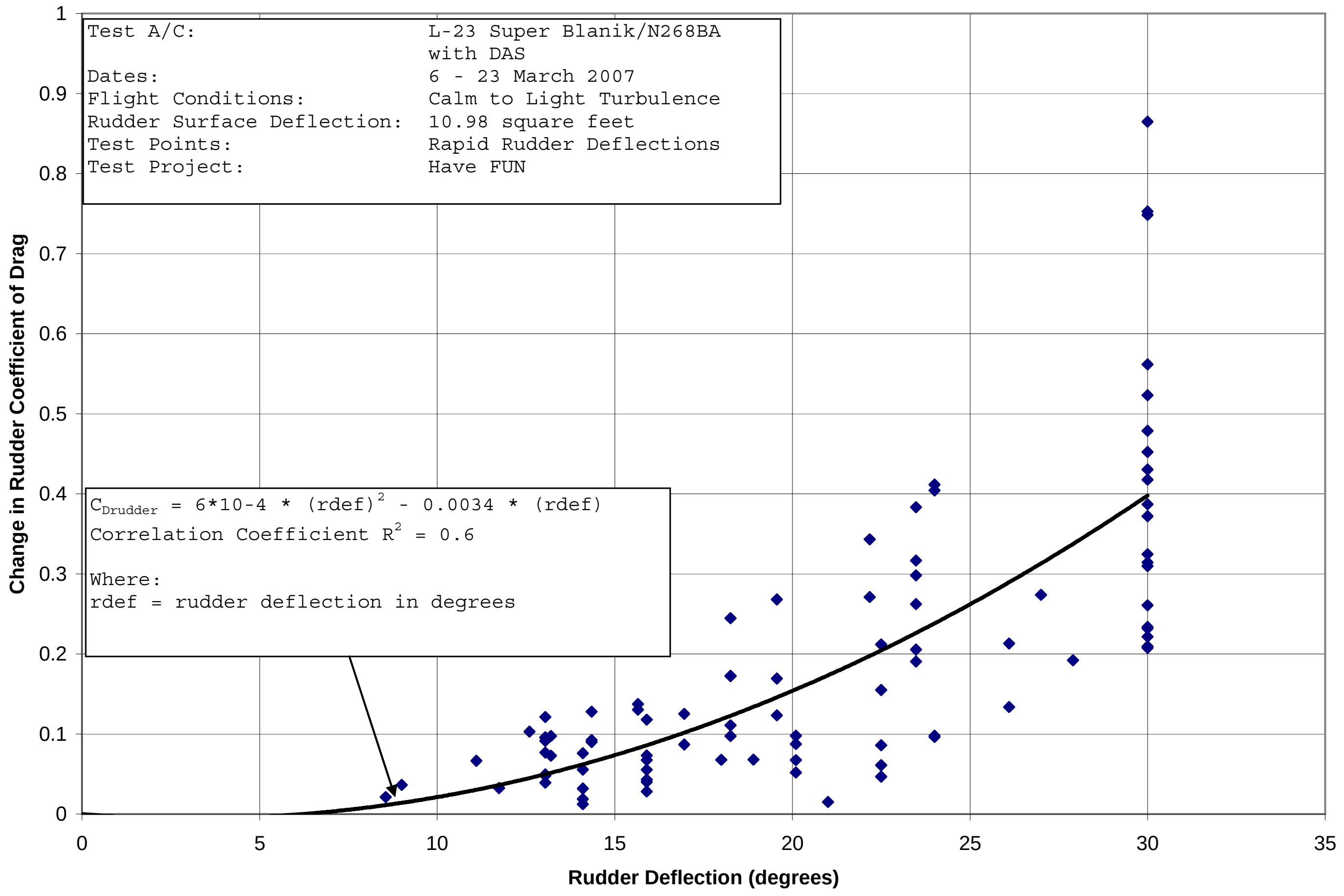


Figure 6: Rudder Deflection Induced Increase in $C_{Drudder}$

Glide, Porpoise, and Dynamic Maneuver Comparison

A comparison of specific energy loss was performed between maneuvering flight and straight glide using the maneuvers described in the previous flight test technique section. Each maneuver was accomplished in still air when available, and yielded the following results. Figures D 4 and D 5 are representative MATLAB® generated graphs that were used for analysis (appendix D). Maneuver duration, maneuver quality, and energy height losses were determined from these graphs and plotted in figure 7. The dynamic soaring maneuver resulted in the greatest loss of specific energy, the porpoise exhibited the second greatest loss, and the straight glide lost the least amount of energy.

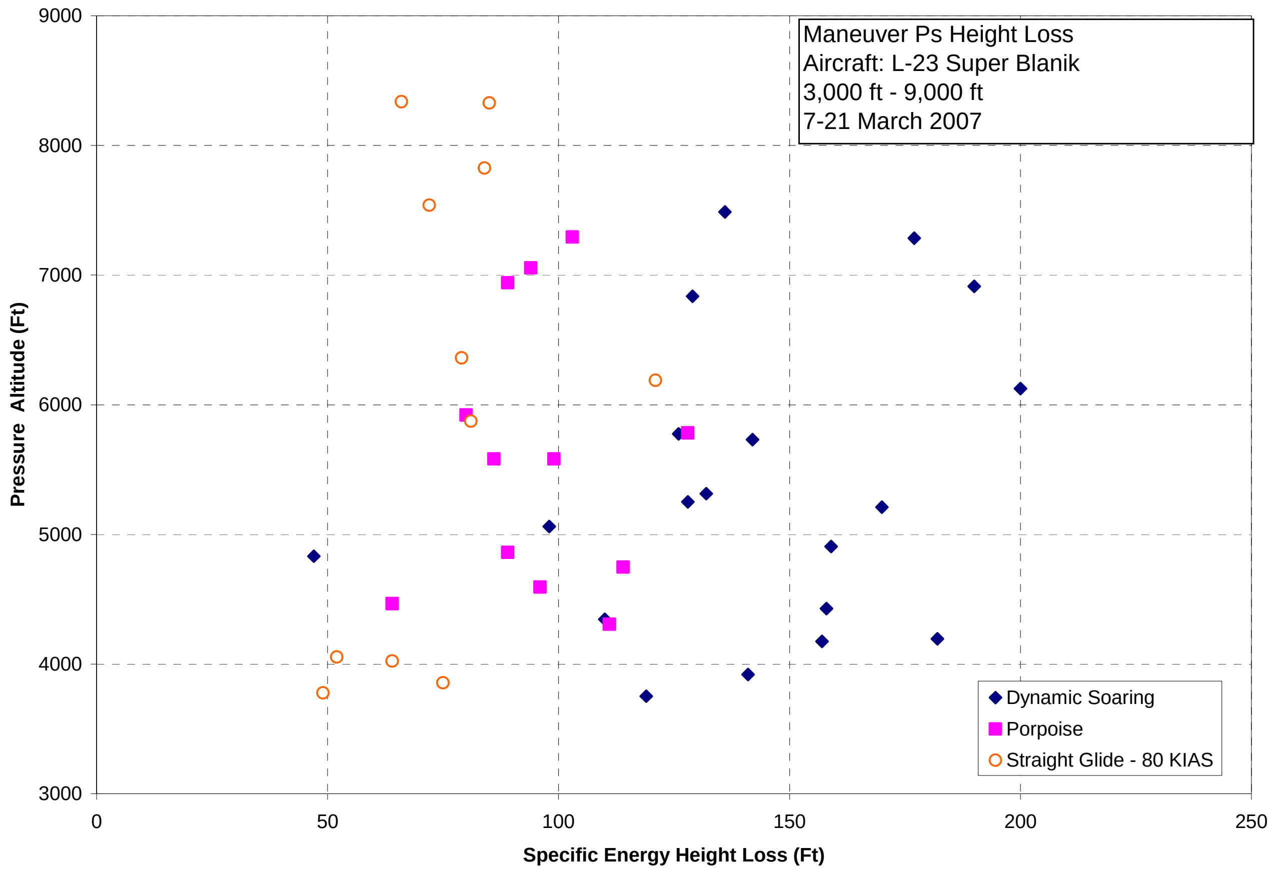


Figure 7: Specific Energy Height Loss Raw Data

Synthesis of Drag Contribution with Comparison of Maneuvers

To synthesize the results of the first two test objectives, the test team performed the following comparison: What energy height loss would the results of objective one predict, and how did that compare to the actual energy height loss measured in objective two? To perform this comparison, the test team used a snap-shot in time and space to calculate the energy height loss predicted at 6000 feet over the course of 17 seconds (the average duration of a dynamic soaring maneuver) using the algorithm and equations presented in appendix E. Calculations were based on the following parameters: during the course of a dynamic soaring maneuver, the average airspeed was 60 KIAS, average load factor was 1.2g, and average alpha was 0.5°. The test team calculated the basic aircraft C_d at 0.5° alpha to be 0.02838 using the drag polar devised by the SENIOR IDS test team (reference 2). Such a C_d would result in an aircraft drag of 55.5 pounds at the 6000 feet, 60 knot condition investigated. Using an average rudder deflection of 10 degrees and an average composite aileron deflection of 9 degrees, the test team determined the C_d (rudder) to be 0.058, and the C_d (aileron) to be 0.02835. When applied to the 6000 foot, 60 knot condition, the surfaces contributed an additional 19.44 pounds of drag an increase of 35 percent over the basic aircraft drag at that condition.

The additional drag due to control surface deflections, the elevated load factor, and increased AOA value associated with the dynamic soaring maneuver predicted a specific energy loss of 95 feet over the course of a 17 second maneuver (appendix E). By comparison, the actual energy loss measured during the dynamic soaring maneuver was 146 feet. Therefore, the drag contribution of aileron and rudder deflection during the dynamic soaring maneuver predicts approximately 65 percent of the energy height loss measured during the maneuver.

Although the HAVE FUN test team was able to use results from control surface drag models to partially predict energy height losses during the dynamic soaring maneuver, about one-third of the actual energy height loss could not be predicted. Therefore, additional factors contribute drag during dynamic soaring. Objective 3 aimed to determine what those contributions were.

Additional Drag Contributors During Dynamic Soaring

The test team was unable to determine the drag contribution of any of the above rates due to the noise in the DAS. The HAVE FUN test plan foreshadowed this particular limitation, stating in the test plan "…excessive noise in the accelerometers could prevent the ability to measure the accelerations required to determine the drag due to control surface deflections". Although the noise did not prevent a determination of drag due to control surface deflections, it may have prevented a determination of drag due to yaw rate, roll rate, and angle-of-attack rate. **Investigate the uncertainty caused by the noise in the DAS. (R1)**[1]

Furthermore, the test team was unable to determine the drag due to sideslip during this test because of the methodology used. Namely, the flight test techniques were designed to achieve the first two objectives. The last objective was written as an ancillary objective, to be met incident to gathering data for the first two. However, objective one maneuvers used instantaneous deceleration readings, whereas the sideslip took time to build up. Therefore, instantaneous deceleration readings did not provide a useful measure of drag due to sideslip. Sideslip may contribute a significant amount of drag during the dynamic soaring maneuver, as it was nearly impossible to fly the maneuver without inducing sideslip. Sideslip may account for much of the drag not explained by the drag contribution of full scale surface deflection. **Investigate the drag contribution of sideslip during the dynamic soaring maneuver. (R2)**

[1] Numerals preceded by an R within parentheses at the end of a sentence correspond to the recommendation numbers tabulated in the Conclusions and Recommendations section of this report

CONCLUSIONS AND RECOMMENDATIONS

Of the three test objectives, the test team determined the drag contribution of aileron and rudder surface deflection, and compared the dynamic soaring maneuver with porpoise and straight glides. The test team did not determine the drag contribution of elevator surface deflection, yaw, yaw rate, roll rate, or angle-of-attack rate. The recommendations drawn from the conclusions, below, are presented in order of priority.

HAVE FUN determined the drag contribution of full scale composite aileron deflections to vary according to the model:

$$C_{D_{aileron}} = 5.0*10^{-5}(\pm 7.0*10^{-5})*adef^{2} + 0.002(\pm 0.0016)*adef$$

Where: *adef* = composite aileron deflection (deg)

Additionally, the full scale rudder deflection contributed drag according to the model:

$$C_{D_{rudder}} = 6.0*10^{-4}(\pm 1.9*10^{-4})*rdef^{2} - 0.0034(\pm 0.0047)*rdef\ .$$

Where: *rdef* = rudder deflection (deg)

Unfortunately, the test team could not determine the drag contribution of sideslip, yaw rate, roll rate, and angle-of-attack rate due to uncertainty in the DAS.

Investigate the uncertainty caused by the noise in the DAS. (R1, page 11)

Furthermore, the test team found that the specific energy height loss in smooth atmospheric conditions was greater in a porpoise maneuver than in a straight glide at comparable airspeeds, and that energy height loss was greatest in the dynamic soaring maneuver. Furthermore, the model of aileron and rudder drag contribution could be used to predict approximately 65 percent of the additional energy height loss measured in the dynamic soaring maneuver versus a straight glide.

During the dynamic soaring maneuver the test team noted significant sideslip, which may account for some of the energy loss not predicted by control surface deflection drag.

Investigate the drag contribution of sideslip during the dynamic soaring maneuver. (R2, page 12)

This page intentionally left blank.

REFERENCES

1. *Test Pilots: Frontiersmen of Flight*, Richard P. Hallion, Smithsonian Press

2. Borror, Sean, Major, *USAF, USAF TPS L-23 Super Blanik Drag Polar and Preliminary Investigation of Dynamic Soaring*, USAF TPS, AFFTC-TIM-04-04, Air Force Flight Test Center, Edwards AFB, California, September 2004.

3. Gordon, Randy, Captain, USAF, *USAF TPS L-23 Shear Wind Observed Optimized Advanced Path Investigation for NASA*, USAF TPS, AFFTC-TIM-06-02, Air Force Flight Test Center, Edwards AFB, California, June 2006.

4. Williams, Michael, 1Lt, USAF, *A Limited Evaluation of Dynamic Soaring Through Thermal Updrafts*, USAF TPS, AFFTC-TIM-06-07, Air Force Flight Test Center, Edwards AFB, California, December 2006.

5. Let, *L-23 Super-Blanik Sailplane Flight Manual*, Do-L23.1011.5, June 1992.

This page intentionally left blank.

Appendix A: Instrumentation and Displays

Sensors

The instrumentation system installed on the Blanik consisted of an Inertial Measurement Unit (IMU), an air data probe and transducers, analog-to-digital converter, a temperature probe and two tablet PC displays. The IMU and analog-to-digital converter were mounted on an adjustable plate and aligned with the centerline of the aircraft. The centerline was defined by the rib running along the top surface of the aft fuselage. The plate was then tilted to align it with the aircraft fuselage reference line. The fuselage reference line was defined by two marks on the side of the glider at the forward and aft ends. The plate was tilted left and right to align with the leading edge of the wing. The air data probe was aligned with the fuselage reference line and centered along the aircraft centerline. All angular measurements were therefore referenced to a body axis coordinate system.

Guidestar GS-111m

An Athena Controls Guidestar 111m (GS-111m) IMU served as the central component in the instrumentation system. The GS-111m used accelerometers, angular rate sensors, GPS, and a magnetometer to compute a full inertial attitude solution. Pitot-static pressures from a nose-mounted 5-hole probe were measured by the GS-111m to determine airspeed, altitude, angle-of-attack (AOA), and angle-of-sideslip (AOS). Pressure transducers on the GS-111m had a dynamic range of ±26,221.9 Pascal for AOA and AOS, and ±16,596 Pascal for dynamic pressure. Total air temperature was measured by a resistive temperature detector (RTD) mounted under the right wing. The RTD voltage was sampled by a 14 bit analog-to-digital input on the GS-111m. Data were output to tablet PCs for flight displays and recording. The GS-111m updated its navigation solution at 50 hertz. The data sampling rate was software selectable with currently available rates of either 25 hertz or 50 hertz. The 50 hertz sampling rate was used for testing. The GS-111m was modified to accept a digital signal from an analog-to-digital converter that was wired to the pressure transducers. This hardware modification consisted of a circuit board housed in a generic black box that could be mounted anywhere in the proximity of the GS-111m and connected to the GS-111m using an RS-232 serial cable. A full description of the GS-111m and the interface control document can be obtained from Athena Controls.

Air Data Probe

An air data probe from Computer Instruments Corporation was used to measure static pressure, total pressure, AOA, and AOS. The initial design called for a constant 0.75 inch outer diameter probe. This was modified by increasing the diameter of the aft end up to 1.25 inch to provide sufficient wall thickness for attachment to the boom. The AOA and AOS measurements were made using a pressure differential, total pressure, and a scale factor. The probe had a scale factor of 4.526366/radian. During a previous project the air data probe was calibrated using a trailing cone.

Resistive Temperature Detector

The resistive temperature detector (RTD) from Computer Instruments Corporation was used to measure total temperature. The platinum RTD had a nominal resistance of 500 Ohm and a scale factor of 0.00385 Ohm/Ohm/degree Celsius. The RTD was powered by an Action Instruments Ultra Slimpak G418-0001. A full description of this device may be obtained by contacting Computer Instruments Corporation. The output voltage of the RTD was sampled by a 14 bit analog-to-digital converter on the GS-111m. During a previous project an ice bath calibration of the RTD connected to the GS-111m resulted in the following relationship between RTD resistance and measured voltage:

R_RTD = 474.0085 Ω + 61.6398 Ω /volt * Voltage

A platinum RTD had a sensitivity curve with a slope of 0.00385 Ohm/Ohm/degree Celsius over the temp range [-10 to +50] deg C. This gave a relationship between RTD resistance and temperature:

T(°C) = -257.3989 C + 0.5148 C/ Ω * R_RTD

Combining these equations gives a relationship between voltage measured by the INU and total air temperature:

T(°C) =(0.5148 * (474.0085 + (Voltage * 61.6398))) - 257.3989

GS-111m Interface

Interface to the GS-111m was made via five serial ports accessible through 51 pin connectors. Each serial port was configured for RS-232 communication at 115.2 kilobits/second. The slow data rate was chosen primarily to ensure reliable communication with the Motion Computing tablet PC used for cockpit data display. Operationally, only ports 1-3 were used.

Tablet PC

The GS-111m serial port 2 was used to communicate with a Motion Computing tablet PC®. The tablet PC displayed flight parameters in the cockpit using NASA developed ThUMP 1.3 software (figure A1) and had the capability to start and stop data logging on the tablet PC. The tablet PC in the front cockpit was connected to the GS-111m using a serial to USB connector cable. Data from the tablet PC in the front cockpit were passed to the tablet PC in the rear cockpit via an ethernet cable.

Data Acquisition

Data recording on the GS-111m was possible but not used. Since tablet PC operation was required for test, the data recording capability of the tablet PC software was used for all data capture. The data recorded on the tablet PC was in binary and comma separated variable formats in two separate files. The pilot in the front cockpit could start and stop data collection in the ThUMP software using the HOS button located at the top of the stick. Data files names were configurable on the tablet PC and Have FUN used the following naming convention: Have Fun_calc_mmddyyyy hh_mm_ss.bin and Have FUN_calc_mmddyyyy hh_mm_ss.csv.

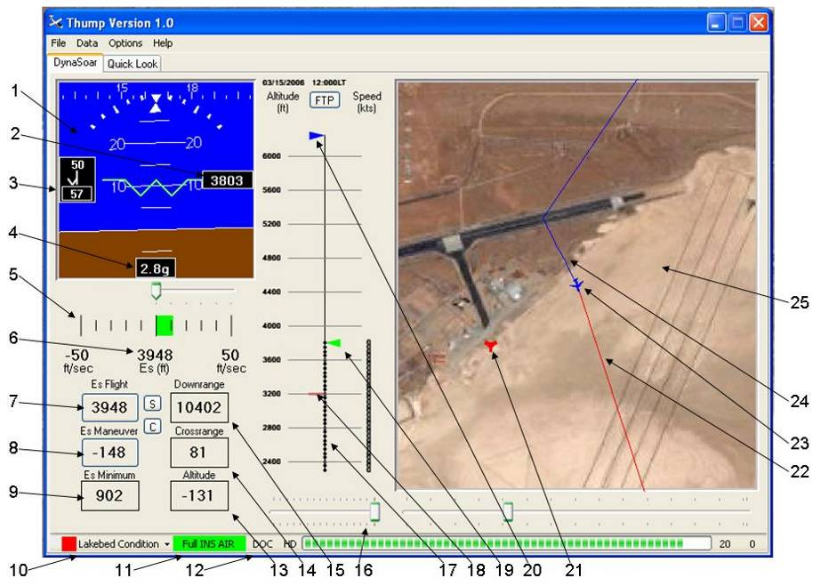


Figure A 1: Tablet PC Display

1	Attitude ball with embedded heading scale and heading bug (bug was set at start of maneuver when the HOS controller was pressed)	14	Cross range distance from start of maneuver (ft)
2	Pressure altitude (ft)	15	Downrange distance from start of maneuver (ft)
3	Airspeed (KIAS)	16	Wind data zoom control
4	Normal load factor	17	Wind data (altitude, direction, speed)
5	Energy rate gauge (ft/sec) (Negative rate displays red, Positive rate displays green)	18	Energy height bingo (ft MSL)
6	Energy height (ft MSL)	19	Pressure altitude (ft)
7	Energy height (ft AGL): Prior to takeoff this button was pressed to zero the energy height. (Below bingo energy the block turns red)	20	Energy height (ft MSL)
8	Energy difference from start of maneuver (ft)	21	Bingo reference point[3]
9	Energy height bingo (ft AGL)[1]	22	Current heading reference line (red)
10	Lakebed status toggle (red/green)[2]	23	Own ship icon
11	EGI/GPS status display	24	Ground track history (blue)
12	DAS status display (Green indicates data is recording)	25	Moving map display with zoom control
13	Altitude from start of maneuver (ft)		

[1] The term “bingo” indicates that a return to base condition has been met. In this case, the minimum energy height needed to prepare for landing would have been reached.

[2] The term “red” lakebeds refers to lakebed conditions that are not conducive to operating the glider from the lakebeds. While “green” lakebeds indicate that the glider can be operated safely from the lakebed.

[3] Energy height bingo reference point (red TACAN symbol) for red lakebed operations was also displayed on the moving map, but it is not in the field of view shown. This point could be set manually under the “Options” – “Energy” tab by entering a latitude and longitude in degrees. During red lakebed operations the energy height bingo was dynamically calculated by assuming a 16:1 L/D ratio to fly from the glider’s current location to the bingo reference point. This allowed the glider to reach the reference point with an energy height equal to 800 ft and 60 KIAS.

Appendix B: Design of Experiments

By eliminating the coordinated turns from the matrix, the team ended up with essentially 3 full factorial matrices, for targeted test conditions

Matrix 1) Full Factorial Cross of 4 Elevator deflections with 5 airspeeds

Matrix 2) Full Factorial Cross of 3 Aileron deflections with 5 airspeeds

Matrix 3) Full Factorial Cross of 3 Rudder deflections with 5 airspeeds.

The test matrix used is on the following pages.

	Maneuver Control Factors				FYI
Count (not priority)	Airspeed	Aileron Deflection	Rudder Deflection	Elevator Deflection	G
1	80	none	none	fixed	1
2	70	none	none	fixed	1
3	60	none	none	fixed	1
4	50	none	none	fixed	1
5	40	none	none	fixed	1
6	40	+2/3	none	fixed	1
7	40	+Full	none	fixed	1
8	40	+1/3	none	fixed	1
9	40	+2/3	none	fixed	1
10	40	+Full	none	fixed	1
11	40	+1/3	none	fixed	1
12	40	none	none	fwd to 0g	0
13	40	none	none	back 1/3	til stable
14	40	none	none	back 2/3	til stable
15	40	none	none	back full	til stable
16	40	none	+1/3	fixed	1
17	40	none	+2/3	fixed	1
18	40	none	+full	fixed	1
19	50	+2/3	none	fixed	1
20	50	+Full	none	fixed	1
21	50	+1/3	none	fixed	1
22	50	+2/3	none	fixed	1
23	50	+Full	none	fixed	1
24	50	+1/3	none	fixed	1
25	50	none	none	fwd to 0g	0
26	50	none	none	back 1/3	til stable
27	50	none	none	back 2/3	til stable
28	50	none	none	back full	til stable
29	50	none	+1/4	fixed	1
30	50	none	+1/2	fixed	1
31	60	+2/3	none	fixed	1
32	60	+Full	none	fixed	1
33	60	+1/3	none	fixed	1
34	60	+2/3	none	fixed	1
35	60	+Full	none	fixed	1
36	60	+1/3	none	fixed	1

37	60	none	none	fwd to 0g	0
38	60	none	none	back 1/3	til stable
39	60	none	none	back 2/3	til stable
40	60	none	none	back full	til stable
41	60	none	+1/4	fixed	1
42	60	none	+1/2	fixed	1
43	70	+2/3	none	fixed	1
44	70	+Full	none	fixed	1
45	70	+1/3	none	fixed	1
46	70	+2/3	none	fixed	1
47	70	+Full	none	fixed	1
48	70	+1/3	none	fixed	1
49	70	none	none	fwd to 0g	0
50	70	none	none	back 1/3	til stable
51	70	none	none	back 2/3	til stable
52	70	none	none	back full	til stable
53	70	none	+1/4	fixed	1
54	70	none	+1/2	fixed	1
55	80	+2/3	none	fixed	1
56	80	+Full	none	fixed	1
57	80	+1/3	none	fixed	1
58	80	+2/3	none	fixed	1
59	80	+Full	none	fixed	1
60	80	+1/3	none	fixed	1
61	80	none	none	fwd to 0g	0
62	80	none	none	back 1/3	til stable
63	80	none	none	back 2/3	til stable
64	80	none	none	back full	til stable
65	80	none	+1/4	fixed	1
66	80	none	+1/2	fixed	1

Notes:
None = zero deflection and held constant throughout the maneuver
Fixed = surface position as required to meet the test point conditions and then held in place for the rest of the maneuver

Some low speed test points were eliminated due to stall considerations. Some high airspeed test points were eliminated due to airframe stress considerations near maneuvering speed. The glider was assumed to be symmetric.

Appendix C: MATLAB® Data Reduction Methodology

The following is the sequence used to reduce the control surface contribution to drag data from a test sortie. It starts with a data file created by the DAS transferred to a MATLAB (Version 7.3.0.267) capable computer. The MATLAB script and function files listed later in this appendix must be in the working directory. Additionally the MATLAB file TPS_data_utility.m (available from Mr. William Gray III at TPS) must be in the same working directory.

Step One: Convert the DAS data .csv file to a MATLAB .mat file

From the MATLAB command prompt, type the command: FUN_data_norm
When the graphical user interface (GUI) opens, click on the 'Select the Subject CSV File' button
Select the DAS data file you wish to convert from the open file GUI
Click on the 'Select the Output MAT File (tpsid_ will be added)' button
Select a location and name for the converted file
Click on the 'GO' button
Once conversion is complete, click on the 'OK' button on the Select File Parameters GUI

Step Two: Find the data

From the MATLAB command prompt, type the command TPS_data_utility
Select the .mat converted data file you wish to view in the open file GUI
Click on the 'OK' button on the Select File Parameters GUI
From the drop down lists on the left side of the TPS data utility GUI, select the following parameters (one per menu):

1. Control surface of interest (aileronDeg, elevatorDeg, or rudderDeg)
2. First control surface to be held constant (one of the two that are not inputs)
3. Second control surface to be held constant (other of the two that are not inputs)
4. rollDeg
5. filtbeta

Click on the 'MAG' button to change the cursor from a pointer to a magnifying glass
Use the click and drag cursor around a suspected control deflection input
Determine data quality by seeing if the non-input controls deflected more than ±2° or if the roll and beta degrees were more than ±2°
If the data point has good quality, note the time and click on the 'Reset Time' button
Search for the next point

Step Three: Collect the data

From the drop down lists on the left side of the TPS data utility GUI, select the following parameters (one per menu):

1. Control surface of maneuver input (aileronDeg, elevatorDeg, or rudderDeg)
2. AxFPS2minusG
3. AzFPS2minusG

4. q
5. filtaoa

Zoom in on a previously located data point
Record the change in x and z acceleration from just after the rapid control input
Record the prior to maneuver angle of attack and dynamic pressure values
Reset the chart and move on to the next data point

Step Four: Determine the drag contributions

Convert the x and z body axis accelerations into total drag with the following equation:

$$D = -Mg*Ax*\text{cosine}(\alpha)-Mg*Az*\text{sine}(\alpha)$$

Where: D = drag
Mg = mass of the glider
Ax = change in x body acceleration
Az = change in z body acceleration
α = angle of attack

Convert the previously calculated drag into a coefficient of drag using the following equation:

$$C_d = D/(q*S)$$

Where: C_d = coefficient of drag
D = drag
q = dynamic pressure
S = surface area of the input control surface

Figure C-1 below is an example of the TPS data utility GUI. This particular case would fall into Step 3 of the above procedure and shows a typical good data point.

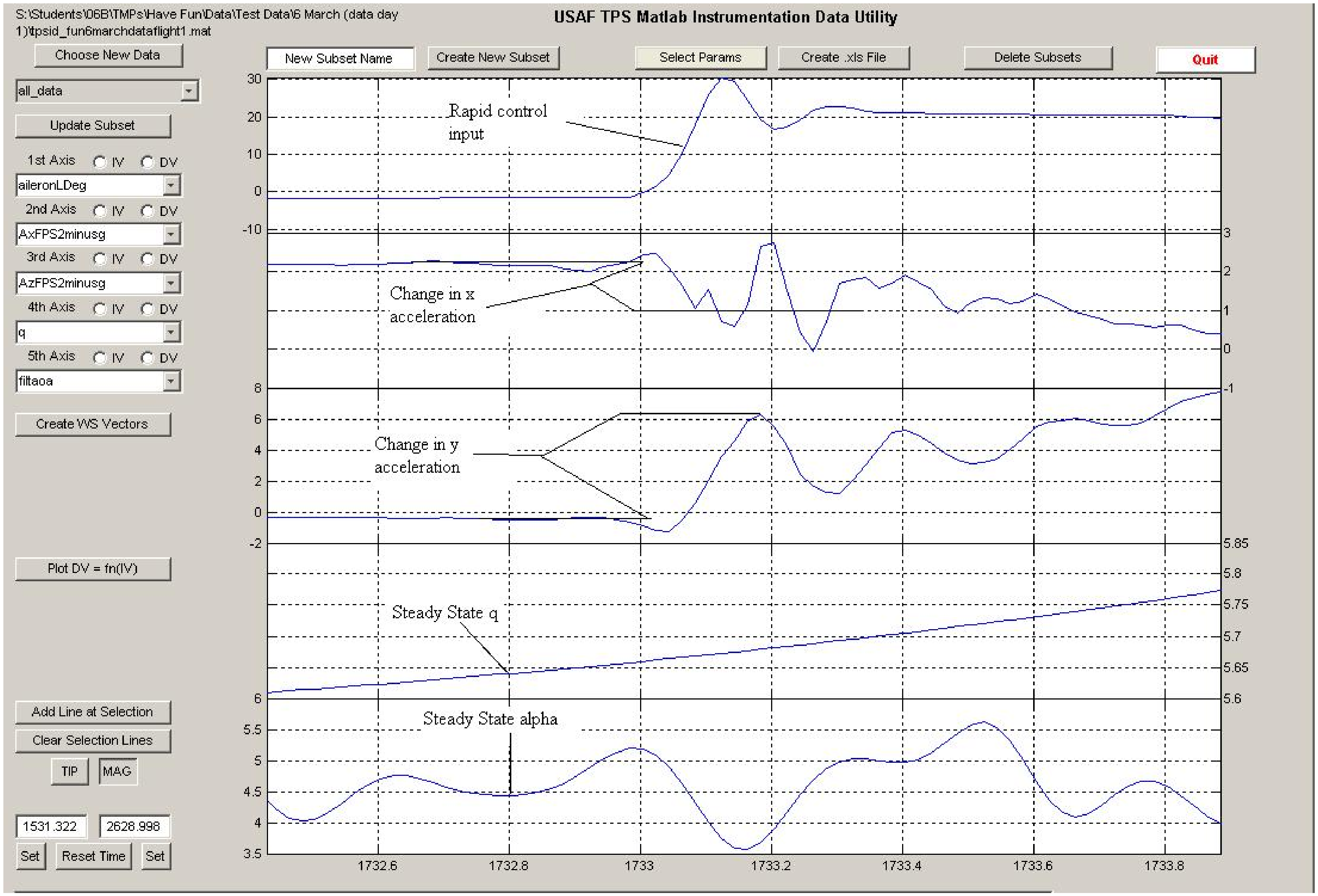


Figure C 1: Data Reduction Example Picture

MATLAB Data Reduction M-File: datanalysis.m

This file calls all of the other scripts and functions used to analyze the data collected during the test flights.

```
%uiopen
%alphaDeg
datfilt; %filters the alpha, beta, IAS, and OAT values.

%gravitycorrection; %subtracts the effects of gravity from the accelerations

gravitycorrection_new;

defnormal; %normalizes the control surface deflections so that full positive
input is 1 and full negative input is -1.

[trueSpeed, q] = atmosdat(altitudeFt,filtOAT,filtAirspeed);

ailLdiff = vdiff(aileronLDeg); %performs a discreet differentiation of the
left aileron deflection versus time

ailRdiff = vdiff(aileronRDeg); %performs a discreet differentiation of the
right aileron deflection versus time

elediff = vdiff(elevatorDeg); %performs a discreet differentiation of the
elevator deflection versus time

ruddiff = vdiff(rudderDeg); %performs a discreet differentiation of the
rudder deflection versus time



%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Aileron Stuff%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%

ailLdatloc = findata(ailLdiff,175,3);

ailRdatloc = findata(ailRdiff,175,3);

aildefplot

j = 1;
numailLpoints = size(ailLdatloc);
while(ailLdatloc(1) > 0 && j<=numailLpoints(1,1))
%    [trueSpeed, q] =
atmosdat(ailLdatloc(j,:),altitudeFt,filtOAT,filtAirspeed);
%    dragdata(i) =
bod2stabconvert(rudLdatloc(j,:),1,zAccelFtPerSecSq,altitudeFt,filtOAT,filtAir
speed,xAccelFtPerSecSq,filtaoa,rudderDeg)
    j=j+1;
end
```

```
j = 1;
numailRpoints = size(ailRdatloc);
while(ailRdatloc(1) > 0 && j<=numailRpoints(1,1))
%    [trueSpeed, q] =
atmosdat(ailRdatloc(j,:),altitudeFt,filtOAT,filtAirspeed);
%    dragdata(i) =
bod2stabconvert(rudLdatloc(j,:),1,zAccelFtPerSecSq,altitudeFt,filtOAT,filtAir
speed,xAccelFtPerSecSq,filtaoa,rudderDeg)
    j=j+1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Elevator Stuff%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

eledatloc = findata(elediff,-400,1);

eleUdatloc = findata(elediff,200,3);

eledefplot

j = 1;
numelepoints = size(eledatloc);
while(eledatloc(1) > 0 && j<=numelepoints(1,1))
%    [trueSpeed, q] =
atmosdat(eledatloc(j,:),altitudeFt,filtOAT,filtAirspeed);
%    dragdata(i) =
bod2stabconvert(rudLdatloc(j,:),1,zAccelFtPerSecSq,altitudeFt,filtOAT,filtAir
speed,xAccelFtPerSecSq,filtaoa,rudderDeg)
    j=j+1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Rudder Stuff%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rudRdatloc = findata(ruddiff,150,3);

rudLdatloc = findata(ruddiff,-150,1);

ruddefplot

j = 1;
numrudLpoints = size(rudLdatloc);
while(rudLdatloc(1) > 0 && j<=numrudLpoints(1,1))
%    [trueSpeed, q] =
atmosdat(rudLdatloc(j,:),altitudeFt,filtOAT,filtAirspeed);
%    dragdata(i) =
bod2stabconvert(rudLdatloc(j,:),1,zAccelFtPerSecSq,altitudeFt,filtOAT,
%    filtAirspeed,xAccelFtPerSecSq,filtaoa,rudderDeg);
    j=j+1;
```

```
end

j = 1;
numrudRpoints = size(rudRdatloc);
while(rudRdatloc(1) > 0 && j<=numrudRpoints(1,1))
%    [trueSpeed, q] =
atmosdat(rudRdatloc(j,:),altitudeFt,filtOAT,filtAirspeed);
%    dragdata(i) =
bod2stabconvert(rudLdatloc(j,:),1,zAccelFtPerSecSq,altitudeFt,filtOAT,filtAir
speed,xAccelFtPerSecSq,filtaoa,rudderDeg)
    j=j+1;
end
```

MATLAB Data Reduction M-File: datfilt.m

This script filters the noisy angle of attack, angle of sideslip, outside air temperature, and indicated airspeed values from the DAS. It uses a Butterworth filter and zero-phase digital filtering to remove noise and phase lag.

```
%function datfilt = datfilt(filterin)
% filter for the HAVE FUN project DAS data

[A,B] = butter(5,10/50);
%filtda = filtfilt(A,B,aileronLDeg);
%filtnx = filtfilt(A,B,xAccelFtPerSecSq);
%figure
%plot(timeTag,filtAlphaDeg)

i = 1;

numpoints = size(alphaDeg);

while(i<=(numpoints(1)-3))
    alphatemp(i) = alphaDeg(i+3);
    betatemp(i) = betaDeg(i+3);
    i = i+1;
end

filtaoa = filtfilt(A,B,alphaDeg);
filtbeta = filtfilt(A,B,betaDeg);

[C,D] = butter(5,2/50);
filtOAT = filtfilt(C,D,OATDegF);
filtAirspeed = filtfilt(C,D,IASFtPerSec);
```

MATLAB Data Reduction M-File: gravitycorrection_new.m

This script subtracts the effects of gravity out of the body x and z accelerations.

```
%This function subtracts out the effects of gravity from the x and z
%accelerations collected from the das.

gravityConst = -32.2;

xAccelFtPerSecSq = xAccelFtPerSecSq - sin((pitchDeg*pi/180)).*gravityConst;
zAccelFtPerSecSq = zAccelFtPerSecSq - cos((pitchDeg*pi/180)).*gravityConst;
```

MATLAB Data Reduction M-File: defnormal.m

```
%This file normalizes the control surface deflections based off of ground
%data taken from the DAS

%Rudder norming
rudderMin = -28.1322;
rudderMax = 15.5579;
rudderTot = rudderMax-rudderMin;

rudderNorm = ((rudderDeg - rudderMin)/rudderTot)*2 - 1;


%elevator norming
elevatorMax = 30.5;
elevatorMin = -31.28;

%check to make sure you have the polarity correct on these values
fltManEleMax = 15;
fltManEleMin = -35;

expandE = (fltManEleMax - fltManEleMin) / (elevatorMax - elevatorMin);
biasE = (expandE * elevatorMax) - fltManEleMax;
eletemp = elevatorDeg * expandE - biasE;

%alieron norming NOTE adjust for differential up vs down deflection.
aileronLMax = 37.5;
aileronLMin = -11.85;
fltManAilMax = 34;
fltManAilMin = -13;

aileronRMax = 31.75;
aileronRMin = -18.295;

expandL = (fltManAilMax - fltManAilMin) / (aileronLMax - aileronLMin);
biasL = (expandL * aileronLMax) - fltManAilMax;
ailLtemp = aileronLDeg * expandL - biasL;

expandR = (fltManAilMax - fltManAilMin) / (aileronRMax - aileronRMin);
biasR = (expandR * aileronRMax) - fltManAilMax;
ailRtemp = aileronRDeg * expandR - biasR;


i = 1;
numpoints = size(aileronLDeg);
while(i<numpoints(1))
    if(elevatorDeg(i)>0)
        elevatorTot(i) = eletemp(i)/fltManEleMax;
    end
    if(elevatorDeg(i)<=0)
        elevatorTot(i) = eletemp(i)/-fltManEleMin;
    end
    if(aileronLDeg(i)>0)
        aileronLTot(i) = ailLtemp(i)/fltManAilMax;
    end
    if(aileronLDeg(i)<=0)
        aileronLTot(i) = ailLtemp(i)/-fltManAilMin;
    end
    if(aileronRDeg(i)>0)
        aileronRTot(i) = ailRtemp(i)/fltManAilMax;
    end
    if(aileronRDeg(i)<=0)
        aileronRTot(i) = ailRtemp(i)/-fltManAilMin;
    end
    i = i+1;
end
```

MATLAB Data Reduction M-File: atmosdat.m

This function calculates and returns the dynamic pressure and true airspeed from the DAS collected altitude, filtered outside air temperature, and filtered airspeed.

```
function [trueSpeed, q] = atmosdat(altitudeFt,filtOAT,filtAirspeed)
%this function calculates the atmospheric data converts airspeed
%accordingly.  It returns the true airspeed and the dynamic pressure (q).


TStdDegF = 59;
i = 1;
numpoints = size(altitudeFt);
j = 1;

while(i  <= numpoints(1))
   deltac(j ,1) = (1 - 0.00000687559 * altitudeFt(i ))^5.2559;
   sigma(j ,1) = deltac(j ,1) / (filtOAT(i) / TStdDegF);
   rho(j ,1) = sigma(j,1) * 0.0023769;
   trueSpeed(j,1) = filtAirspeed(i,1)/(sigma(j,1)^(1/2));
   q(j,1) = 1/2*rho(j,1)*filtAirspeed(i)^2;
   i = i+1;
   j = j+1;
end
```

MATLAB Data Reduction M-File: vdiff.m

This function discretely differentiates a data vector. In this project it was used to get deflection per second from the deflection position vectors. The returned values were used to determine where test inputs were made.

```
function [vectordiff] = vdiff(invector)

%This function takes in a vector to be differentiated and outputs it vector
%derivative.  The function does the differentiation discretely by doing
%change in value divided by change in time.


vecsize = size(invector);
timedif = .02;
i = 2;

vectordiff(1,1) = 0;

while(i<vecsize(1))

    vectordiff(i) = invector(i) - invector(i-1);
    vectordiff(i) = vectordiff(i)/timedif;

    i = i+1;
end
```

MATLAB Data Reduction M-File: findata.m

This function searches an input vector for the limit and the indicated condition. It returns the start and end values of a found data point.

```
function [foundata] = findata(invector, lim, condition)

%This function searches the input vector and returns the location of any
%value that meets the condition.  Condition 1 is less than, condition 2 is
%equal, and condition 3 is greater than.

i = 1;
j = 1;
datloc = 0;

foundata = [0,0];

vecsize = size(invector);

if(condition == 1)
    while(i<vecsize(2))
        if(invector(i) < lim)
            datloc(1,j) = i;
            j = j + 1;
        end
        i = i + 1;
    end
end

if(condition == 2)
    while(i<vecsize(2))
        if(invector(i) == lim)
            datloc(1,j) = i;
            j = j + 1;
        end
        i = i + 1;
    end
end

if(condition == 3)
    while(i<vecsize(2))
        if(invector(i) > lim)
            datloc(1,j) = i;
            j = j + 1;
        end
        i = i + 1;
    end
end

k = 2;
l = 1;

while(k<j)
    if(not((datloc(k) - 1) == datloc(k-1)))
```

```
        newpoint(l) = k;
        l = l+1;
    end
    k = k+1;
end

k = 1;
m = 1;

while(k<l)
    foundata(k,1) = datloc(newpoint(k))-15;
    foundata(k,2) = datloc(newpoint(k))+50;
    k = k+1;
end
```

MATLAB Data Reduction M-File: aildefplot.m

This script plots the aileron deflections, body x accelerations, and body z accelerations found in the data.

```
%plots the aileron deflections found in the findata function to view their
%quality.

numpointsL = size(ailLdatloc);

i = 1;

while(ailLdatloc(1)>0 && i<=numpointsL(1))
    figure
    plot(aileronLTot(ailLdatloc(i,1):ailLdatloc(i,2)))
%     hold on
    title(['Left Aileron #', num2str(i)])
    figure
    plot(xAccelFtPerSecSq(ailLdatloc(i,1):ailLdatloc(i,2)))
    title('x acceleration')
    figure
    plot(zAccelFtPerSecSq(ailLdatloc(i,1):ailLdatloc(i,2)))
    title('z acceleration')
    ailLspeed(i,1) = trueSpeed(ailLdatloc(i,1));
    ailLq(i,1) = q(ailLdatloc(i,1));
    i = i+1;
end

numpointsR = size(ailRdatloc);

i = 1;

while(ailRdatloc(1)>0 && i<=numpointsR(1))
    figure
    plot(aileronRTot(ailRdatloc(i,1):ailRdatloc(i,2)))
%     hold on
    title(['Right Aileron #', num2str(i)])
    figure
    plot(xAccelFtPerSecSq(ailRdatloc(i,1):ailRdatloc(i,2)))
    title('x acceleration')
    figure
    plot(zAccelFtPerSecSq(ailRdatloc(i,1):ailRdatloc(i,2)))
    title('z acceleration')
    ailRspeed(i,1) = trueSpeed(ailRdatloc(i,1));
    ailRq(i,1) = q(ailRdatloc(i,1));
    i = i+1;
end
```

MATLAB Data Reduction M-File: eledefplot.m

This script plots the elevator deflections, body x accelerations, and body z accelerations found in the data.

```
%plots the elevator deflections found in the findata function to view their
%quality.

numpoints = size(eledatloc);


i = 1;

while(eledatloc(1)>0 && i<=numpoints(1))
    figure
    plot(elevatorTot(eledatloc(i,1):eledatloc(i,2)))
%    hold on
    title(['Elevator down #', num2str(i)])
    figure
    plot(xAccelFtPerSecSq(eledatloc(i,1):eledatloc(i,2)))
    title('x acceleration')
    figure
    plot(zAccelFtPerSecSq(eledatloc(i,1):eledatloc(i,2)))
    title('z acceleration')
    elespeed(i,1) = trueSpeed(eledatloc(i,1));
    eleq(i,1) = q(eledatloc(i,1));
    i = i+1;
end



numpointsU = size(eleUdatloc);


i = 1;

while(eleUdatloc(1)>0 && i<=numpointsU(1))
    figure
    plot(elevatorTot(eleUdatloc(i,1):eleUdatloc(i,2)))
%    hold on
    title(['Elevator up #', num2str(i)])
    figure
    plot(xAccelFtPerSecSq(eleUdatloc(i,1):eleUdatloc(i,2)))
    title('x acceleration')
    figure
    plot(zAccelFtPerSecSq(eleUdatloc(i,1):eleUdatloc(i,2)))
    title('z acceleration')
    eleUspeed(i,1) = trueSpeed(eleUdatloc(i,1));
    eleUq(i,1) = q(eleUdatloc(i,1));
    i = i+1;
end
```

MATLAB Data Reduction M-File: ruddefplot.m

This script plots the rudder deflections, body x accelerations, and body z accelerations found in the data.

```
%plots the rudder deflections found in the findata function to view their
%quality.

numpointsL = size(rudLdatloc);
numpointsR = size(rudRdatloc);
i = 1;

while(rudRdatloc(1)>0 && i<=numpointsR(1))
    figure
    plot(rudderNorm(rudRdatloc(i,1):rudRdatloc(i,2)))
%     hold on
    title(['Right Aileron #', num2str(i)])
    figure
    plot(xAccelFtPerSecSq(rudRdatloc(i,1):rudRdatloc(i,2)))
    title('x acceleration')
    figure
    plot(zAccelFtPerSecSq(rudRdatloc(i,1):rudRdatloc(i,2)))
    title('z acceleration')
    hold off
    rudRspeed(i,1) = trueSpeed(rudRdatloc(i,1));
    rudRq(i,1) = q(rudRdatloc(i,1));
    i = i+1;

end

i = 1;

while(rudLdatloc(1)>0 && i<=numpointsL(1))
    figure
    plot(rudderNorm(rudLdatloc(i,1):rudLdatloc(i,2)))
%     hold on
    title(['Right Aileron #', num2str(i)])
    figure
    plot(xAccelFtPerSecSq(rudLdatloc(i,1):rudLdatloc(i,2)))
    title('x acceleration')
    figure
    plot(zAccelFtPerSecSq(rudLdatloc(i,1):rudLdatloc(i,2)))
    title('z acceleration')
    rudLspeed(i,1) = trueSpeed(rudLdatloc(i,1));
    rudLq(i,1) = q(rudLdatloc(i,1));
    i = i+1;

end
```

MATLAB Data Reduction M-File: FUN_data_norm.m

This function takes the comma separated values file that the DAS creates and converts it to a MATLAB data file. The vast majority of this function was written by Mr. William Gray III. The modifications made by Capt Geoffrey Bowman for the Have FUN TMP are listed in the comments.

```
function varargout = FUN_data_norm(varargin)
%FUN_data_norm M-file for FUN_data_norm.fig
%
%Just type 'FUN_data_norm' at the >> prompt to run the GUI.
%
%GUI Options:
%   Press "Select the Subject CSV File" to choose the CSV file with the
%   data.
%This file may come from ILIAD (T-38 or F-16) or the C-12 card reader
%utility. The file need not have text headers but if it does, it will
%identify the 'Delta_Irig' column from ILIAD files and use this time for
%the time data. C-12 files contain two types of date/time group and this
%program uses the ASCII time column to create a delta time column in
%seconds labeled 'DELTA_TIME.'
%
%   The processed output is re-formatted as a  MATLAB structure containing
%several fields. If requested by checking 'Create Separate Event
%Structures', a separate structure will be created for the duration of each
%event.* All of these structures are saved into a single MAT file with the
%specified name and location ('tpsid_' is attached to the beginning of the
%file name to signify that it is compatible with the data review utility.
%Each structure has the following fields:
%   date: The date and time the file was processed in MATLAB vector format.
%   subset: 'all_data' is all of the data. Each of the event segments is
%       saved as 'EC_XXX' where XXX is the event number.
%   time_vec: The data vector containing the time in seconds. If no time is
%       available or identified, this vector will contain the sample (row)
%       number.
%   event_col: If known, the column containing event numbers. If unknown,
%       this variable will contain -1.
%   vec_names: The vector (header names) corrected to conform to MATLAB
%       variable standards.
%   data: The data itself. Non-numerical data is converted to NaN.
%   num_vecs: The number of data vectors. samples: The total number of time
%   slices. note: A text note of unlimited length. samp_rate_ave: The
%   average sample rate. displayvecs: The five columns pre-selected for
%   display in the data
%       tool.
%   ylims: The YLims for use in the data tool.
%
%   If desired, the tool will automatically open the data review utility.
%the user must still select the desired file when the utility opens.
%
%*If an event number is not contiguous (occurs more than once in the data
%stream) all of the data associated with that number will be in one vector.
%This is a place for later improvement.
```

```
% Last Modified by GUIDE v2.5 10-Apr-2006 15:15:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @FUN_data_norm_OpeningFcn, ...
    'gui_OutputFcn',  @FUN_data_norm_OutputFcn, ...
    'gui_LayoutFcn',  [], ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before FUN_data_norm is made visible.
function FUN_data_norm_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for FUN_data_norm
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);




% --- Outputs from this function are returned to the command line.
function varargout = FUN_data_norm_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in select_file.
function select_file_Callback(hObject, eventdata, handles)

[handles.CSVFileName,handles.CSVPathName] = uigetfile('*.csv','Select the CSV
File');
handles.CSV_file = [handles.CSVPathName handles.CSVFileName];
set(handles.csvfile,'String',handles.CSV_file);

guidata(hObject, handles);



% --- Executes on button press in go.
function go_Callback(hObject, eventdata, handles)
```

```
%No point in going on if there is no data or target.
if ~isfield(handles,'CSV_file') | ~isfield(handles,'MAT_file')
    warndlg('Enter CSV and MAT names first!','Missing subset name(s)...')
    return
end

%Sort out the data.
csv_to_structs(handles.CSV_file,handles.MAT_file,1,...
    get(handles.notein,'String'), get(handles.createSES,'Value'));

%Send to the data review function
if get(handles.openQDR,'Value') == 1
    TPS_data_utility; end


% --- Executes on button press in putfile.
function putfile_Callback(hObject, eventdata, handles)

%Get a file name and make sure it starts with tpsid_
[handles.MATFileName,handles.MATPathName] =...
    uiputfile('tpsid_*.mat','Select the CSV File',handles.CSVPathName);
checker = strfind(handles.MATFileName,'tpsid_');
if isempty(checker) || checker ~= 1
    handles.MATFileName = ['tpsid_' handles.MATFileName];
end
handles.MAT_file = [handles.MATPathName handles.MATFileName];
set(handles.matfile,'String',handles.MAT_file);

guidata(hObject, handles);


% --- Executes on button press in quit.
function quit_Callback(hObject, eventdata, handles)

close(handles.figure1)


function csv_to_structs(csv_data,mat_data,split,note,events)

%The BIG function...

stat = msgbox(['Reading CSV file for conversion...'],'Progress');

%Find the number of rows in the file
fid_in = fopen(csv_data);
firstcol = textscan(fid_in, '%s%*[^\n]','delimiter',',');
fclose(fid_in);
totrows=size(firstcol{1,1},1)-1;
clear firstcol;

%In this application, split is always 1, but in the future it might be
%advantageous to let the user choose how many rows per data read.
if split == 1
```

```
    split = round((totrows-1)/4);
    if split < 100
        split=100;
    else
        if split > 20000
            split = 20000;
        end
    end
end
iterations=round((totrows-1)/split+.5); %The number of data reads required

%Get to work... all_data is the structure that will contain the data. We
%start with the full data structure.

all_data.date = mat2str(clock);
all_data.subset = 'all_data';

%open the csv file for reading
fid_in = fopen(csv_data);

%Read in the first row of data (assumed vector names)
C = textscan(fid_in, '%s',1, 'delimiter','\r');
d_names = strread(C{1,1}{1,:},'%s','delimiter',',')'; %read vector names

%Initialize some variables
save_names = []; %This will be a vector of column names
ts_format = []; %This will be a string of formats used to read data rows
vects = 0;
all_data.time_vec = -1;
all_data.event_col = -1;
all_data.time_vec = -1;
time_format = 'unknown';
errorind = 'none';

%This section creates the information necessary to successfully read data
%from both ILIAD and C12 files. This is done one column header at a time.

%modified to support have fun data files 3/5/07

for j=1:size(d_names,2)
    %First, ensure that the column header is a valid MATLAB variable name
    expression = ['isvarname(''',cell2mat(d_names(j)),''')'];
    isvar = eval(expression);
    if isvar ~= 1 %NOT a valid name...
        %Replace all characters that make a name not a variable name with
        %an underscore
        d_names{j} = regexprep(cell2mat(d_names(j)),'^\d|\W','_');
    end
    %handle known column names to extract times and events
    switch d_names{j}
        case {'timeTag'} %HAVE FUN telemetry from instrumented Blanik
            vects=vects+1;
            ts_format=[ts_format '%f '];
            save_names{vects} = d_names{j};
            all_data.time_vec = vects;
        case {'telem_time__ASCII_'} %C12 telemetry time; note that this
            %turns one column of data into four.
            ts_format=[ts_format '%f %f %f %f '];
            vects = vects + 1; save_names{vects} = [d_names{j} '_day'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_hr'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_min'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_sec'];
            time_format = 'C12';
            errorind = '-1.#IND00'; %C12 output uses this for numerical
errors
        case {'TIME__ASCII_'} %C12 time (used for running time DELTA_TIME);
            %note that this turns one column of data into four.
            ts_format=[ts_format '%f %f %f %f '];
            vects = vects + 1; save_names{vects} = [d_names{j} '_day'];
            dayvec = vects;
            vects = vects + 1; save_names{vects} = [d_names{j} '_hr'];
            hrvec = vects;
            vects = vects + 1; save_names{vects} = [d_names{j} '_min'];
            minvec = vects;
            vects = vects + 1; save_names{vects} = [d_names{j} '_sec'];
            secvec = vects;
            time_format = 'C12';
            errorind = '-1.#IND00';
        case {'IRIG_TIME'} %ILIAD IRIG time (T38 and F16); this turns one
            %column into five.
            ts_format=[ts_format '%f %f %f %f %f '];
            vects = vects + 1; save_names{vects} = [d_names{j} '_day'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_hr'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_min'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_sec'];
            vects = vects + 1; save_names{vects} = [d_names{j} '_dsec'];
            time_format = 'ILIAD';
            errorind = {'-1.#IND','1.#INF'}; %The ILIAD output for numerical
errors
        case ('Delta_Irig') %Identifies the running seconds column from ILIAD
            %output files.
            vects=vects+1;
            ts_format=[ts_format '%f '];
            save_names{vects} = d_names{j};
            all_data.time_vec = vects;
        case ('EVENT_COUNTER') %Identifies the event column from ILIAD output
            vects=vects+1;
            ts_format=[ts_format '%f '];
            save_names{vects} = d_names{j};
            all_data.event_col = vects;
        otherwise
            vects=vects+1; %All other columns unless...
            %the name includes 'EVENT' then it is identified as the event
            %column
            if ~isempty(strfind(cell2mat(d_names(j)),'EVENT'))
                all_data.event_col = vects; end
            ts_format=[ts_format '%f '];
            save_names{vects} = d_names{j};
    end
end

%Some structure elements can be created directly.
```

```
all_data.vec_names = save_names;
all_data.data = zeros(totrows,vects);
all_data.num_vecs = vects;
all_data.samples = totrows;
all_data.note = note;

%For the rest of the structure elements...

%Data must be read into the structure in pieces so that the typically large
%files do note choke the computer.
endrow=0;
for i=1:iterations

    %Read in a slug of data. Multiple delimiters are necessary for C12
    %output but only the ',' delimiter is used in ILIAD output.
    C = textscan(fid_in, ts_format, split, 'delimiter', '[,: ]',...
        'multipleDelimsAsOne', 1,... %Needed for C12 data
        'treatAsEmpty', errorind); %Depends on the data source

    startrow=endrow+1;
    endrow=endrow+length(C{1,1});

    %Append the numerical data to the data already saved.
    all_data.data(startrow:endrow,:)=cell2mat(C);

    %Let the user know the status of the read.
    if ishandle(stat); close(stat); end;
    stat = msgbox(['Creating Matlab structure from all data, '...
        num2str((i)/iterations*100) '% complete.'],'Progress...');

end

%The data is all read now, so we can close the CSV file.
fclose(fid_in);

%C12 files need a time column and everything else needs some independent
%variable column for the data review utility.
if all_data.time_vec == -1
    switch time_format
        %No pre-identified running time column will cause the creation of a
        %vector of the row count
        case ('unknown')
            all_data.vec_names = ['count' all_data.vec_names];
            counts = [1:all_data.samples]';
            all_data.data = [counts all_data.data];
            [all_data.samples all_data.num_vecs] = size(all_data.data);
            all_data.time_vec = 1;
            %C12 files have enough data in the ASCII TIME column to create
            %a running time independent variable.
        case ('C12')
            dtime = all_data.data(:,dayvec)*24*60*60 +...
                all_data.data(:,hrvec)*60*60 +...
                all_data.data(:,minvec)*60 +...
                all_data.data(:,secvec);
            dtime = dtime - min(dtime);
            all_data.vec_names = ['DELTA_TIME' all_data.vec_names];
            all_data.data = [dtime all_data.data];
            [all_data.samples all_data.num_vecs] = size(all_data.data);
            all_data.time_vec = 1;
    end
    all_data.event_col = all_data.event_col + 1; %The time vector is added
    %as the first column so the ID for the event column must be changed.
end

%Calculate the average sample rate. This data is not currently used by
%might be useful for future data processing in the data review GUI.
samp_rate = mean(all_data.data(2:end,all_data.time_vec)...
    - all_data.data(1:end-1,all_data.time_vec));
all_data.samp_rate_ave = samp_rate;

%Here we create new data based upon the instrumented data

%Add Filtered AOA--comments on procedure here!
    %create data vector
alphaDeg = all_data.data(:,strmatch('alphaDeg',all_data.vec_names));
    %timeshift vector 60ms
alphaDeg = [alphaDeg(3:end);alphaDeg(end);alphaDeg(end)];
    %create and apply filter to make the new data
[A,B] = butter(5,10/50);
filtaoa = filtfilt(A,B,alphaDeg);
    %add the data name to the vector names
all_data.vec_names = [all_data.vec_names 'filtaoa'];
    %add the filtered data to the data
all_data.data = [all_data.data filtaoa];

%Add Filtered beta
betaDeg = all_data.data(:,strmatch('betaDeg',all_data.vec_names));
betaDeg = [betaDeg(3:end);betaDeg(end);betaDeg(end)];
[A,B] = butter(5,10/50);
filtbeta = filtfilt(A,B,betaDeg);
all_data.vec_names = [all_data.vec_names 'filtbeta'];
all_data.data = [all_data.data filtbeta];

%Add Filtered OAT
OATDegF = all_data.data(:,strmatch('OATDegF',all_data.vec_names));
[A,B] = butter(5,1/50);
filtOAT = filtfilt(A,B,OATDegF);
all_data.vec_names = [all_data.vec_names 'filtOAT'];
all_data.data = [all_data.data filtOAT];

%Add Filtered IAS
IASFtPerSec = all_data.data(:,strmatch('IASFtPerSec',all_data.vec_names));
[A,B] = butter(5,1/50);
filtAirspeed = filtfilt(A,B,IASFtPerSec);
all_data.vec_names = [all_data.vec_names 'filtAirspeed'];
all_data.data = [all_data.data filtAirspeed];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Normalize the control deflections %%
%% Mod by Capt Geoff Bowman          %%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


rudderDeg = all_data.data(:,strmatch('rudderDeg',all_data.vec_names));
    %do this for each vector you need
ruddefnorm = rudnorm(rudderDeg);
all_data.vec_names = [all_data.vec_names 'ruddefnorm'];
all_data.data = [all_data.data ruddefnorm];

elevatorDeg = all_data.data(:,strmatch('elevatorDeg',all_data.vec_names));
    %do this for each vector you need
eledefnorm = elenorm(elevatorDeg);
all_data.vec_names = [all_data.vec_names 'eledefnorm'];
all_data.data = [all_data.data eledefnorm];

aileronLDeg = all_data.data(:,strmatch('aileronLDeg',all_data.vec_names));
    %do this for each vector you need
ailLdefnorm = ailLnorm(aileronLDeg);
all_data.vec_names = [all_data.vec_names 'ailLdefnorm'];
all_data.data = [all_data.data ailLdefnorm];

aileronRDeg = all_data.data(:,strmatch('aileronRDeg',all_data.vec_names));
    %do this for each vector you need
ailRdefnorm = ailRnorm(aileronRDeg);
all_data.vec_names = [all_data.vec_names 'ailRdefnorm'];
all_data.data = [all_data.data ailRdefnorm];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Caculate the true airspeed and dynamic pressure from atmospheric data %%
%% mod made by Capt Geoff Bowman                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

altitudeFt = all_data.data(:,strmatch('altitudeFt',all_data.vec_names));
filtOAT = all_data.data(:,strmatch('filtOAT',all_data.vec_names));
filtAirspeed = all_data.data(:,strmatch('filtAirspeed',all_data.vec_names));
    %do this for each vector you need
[trueSpeed, q] = atmosdat(altitudeFt,filtOAT,filtAirspeed);
all_data.vec_names = [all_data.vec_names 'trueSpeed'];
all_data.data = [all_data.data trueSpeed];
all_data.vec_names = [all_data.vec_names 'q'];
all_data.data = [all_data.data q];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Correct the accelerations to take out the effects of gravity %%
%% mod made by Capt Geoff Bowman                                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xAccelFtPerSecSq =
all_data.data(:,strmatch('xAccelFtPerSecSq',all_data.vec_names));
zAccelFtPerSecSq =
all_data.data(:,strmatch('zAccelFtPerSecSq',all_data.vec_names));
pitchDeg = all_data.data(:,strmatch('pitchDeg',all_data.vec_names));
    %do this for each vector you need
[AxFPS2minusg, AzFPS2minusg] = gravitycorrection(xAccelFtPerSecSq,
zAccelFtPerSecSq, pitchDeg);
```

```
all_data.vec_names = [all_data.vec_names 'AxFPS2minusg'];
all_data.data = [all_data.data AxFPS2minusg];
all_data.vec_names = [all_data.vec_names 'AzFPS2minusg'];
all_data.data = [all_data.data AzFPS2minusg];




%Add additional filtered or calculated values here (such as TAS)
%inputDataName =
all_data.data(:,strmatch('inputDataName',all_data.vec_names));
    %do this for each vector you need
%newDataName = fn(inputDataName1,inputDataName2);
%all_data.vec_names = [all_data.vec_names 'newDataName'];
%all_data.data = [all_data.data newDataName];




%do not change anything after this!

if ishandle(stat); close(stat); end %Close any open status windows

%The user is asked which vectors will serve as the default display vectors
%in the data utility tool.
[displayvecs,ok] = listdlg('PromptString',...
    'Select up to five parameters for the plots:',...
    'SelectionMode','multiple',...
    'ListString',all_data.vec_names,...
    'ListSize',[240 600],...
    'Name','Select Plot Parameters');

if ok==0 return; end

displayvecs = [displayvecs 2 3 4 5 6]; %Add a buffer in case fewer than
%five vectors were selected
all_data.displayvecs = displayvecs(1:5); %Create the vector in all_data

%Create a 2xcolumns vector of YLims for later charting
padding = .025*(max(all_data.data)-min(all_data.data));
all_data.ylims = [min(all_data.data)-padding; max(all_data.data)+padding];
%k will be 0 if the YLims are the same...
k = 0 == (all_data.ylims(2,:)-all_data.ylims(1,:));
%Then it is used to make YLims for vectors with only one value
all_data.ylims(1,:) = all_data.ylims(1,:) - k;
all_data.ylims(2,:) = all_data.ylims(2,:) + k;

%The all_data structure is done, time to save it to the MAT file.
eval(['save ''' mat_data ''' all_data']);

%If there are unique event numbers and the user desires, create one
%additional structures for each event number
if all_data.event_col > 0 & events == 1
    %Find the event counter values
    event_data = eval(['all_data.data(:,' num2str(all_data.event_col) ');']);
    a = unique(event_data);
    event_counters = a(find(a>=0)); %This is the vector of event counters
    %Create the event structures one at a time
    for i = 1:length(event_counters)
        counter = event_counters(i);
        stat = msgbox(['Creating structures for counter '...
            num2str(counter)], 'Progress...');
        indexes = find(counter==event_data); %Where the event counter data is
        if length(indexes) > 1
            event_str = ['EC_' num2str(counter, '%03g')];
            eval([event_str '=all_data;']);
            eval([event_str '.data = all_data.data(indexes,:);']);
            eval([event_str '.subset=''Event ' num2str(counter) ''';']);
            eval([event_str '.samples=length(indexes);']);
            eval([event_str...
                '.note=''Auto-Generated'';']);
            samp_rate = mean(eval([ event_str
'.data(2:end,all_data.time_vec)'])...
                - eval([ event_str '.data(1:end-1,all_data.time_vec)']));
            eval([event_str '.samp_rate_ave = samp_rate;']);
            mins = eval(['min(' event_str '.data);']);
            maxs = eval(['max(' event_str '.data);']);
            padding = .025*(maxs-mins);
            eval([event_str '.ylims = [mins-padding; maxs+padding];']);
            eval(['k = 0 == (' event_str '.ylims(2,:)-' event_str
'.ylims(1,:));']);
            eval([event_str '.ylims(1,:) = ' event_str '.ylims(1,:) - k;']);
            eval([event_str '.ylims(2,:) = ' event_str '.ylims(2,:) + k;']);
            eval(['save ''' mat_data ''' ' event_str ' -append']);
            if ishandle(stat); close(stat); end;
        else
            %Sometimes there is only one row for an event counter--these
            %event counters are skipped.
            stat = msgbox(['Event ' num2str(counter)...
                ' subset was not automatically generated; only one row.'],...
                'Progress...');
            pause(5); if ishandle(stat); close(stat); end;
        end
    end
end

stat = msgbox(['Conversion complete!']);
pause(5);
if ishandle(stat); close(stat); end
```

Appendix D: Supporting Flight Test Data

Objective 1

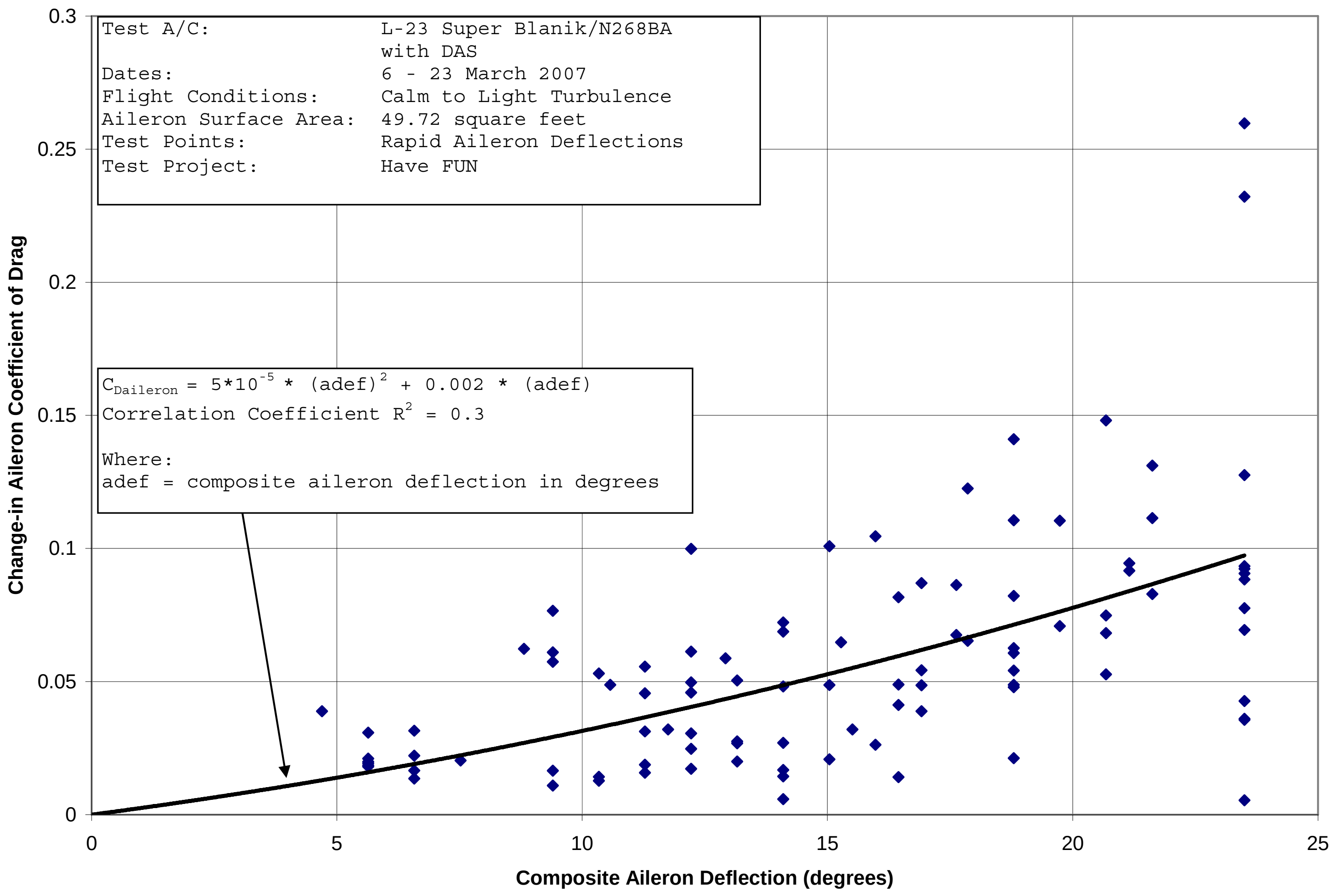


Figure D 1: Aileron Deflection Induced Increase in C_d

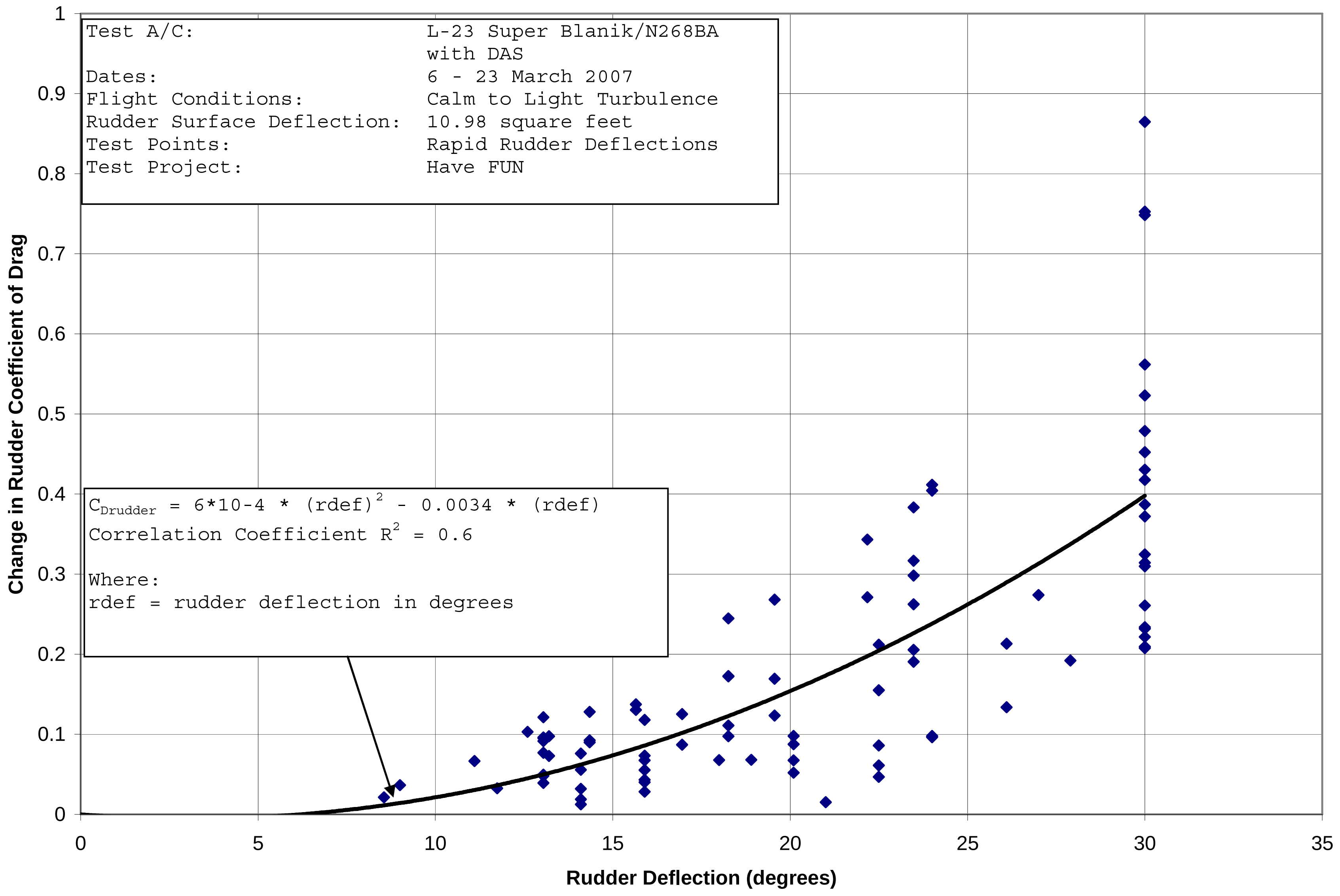


Figure D 2: Rudder Deflection Induced Increase in C_d

Objective 2

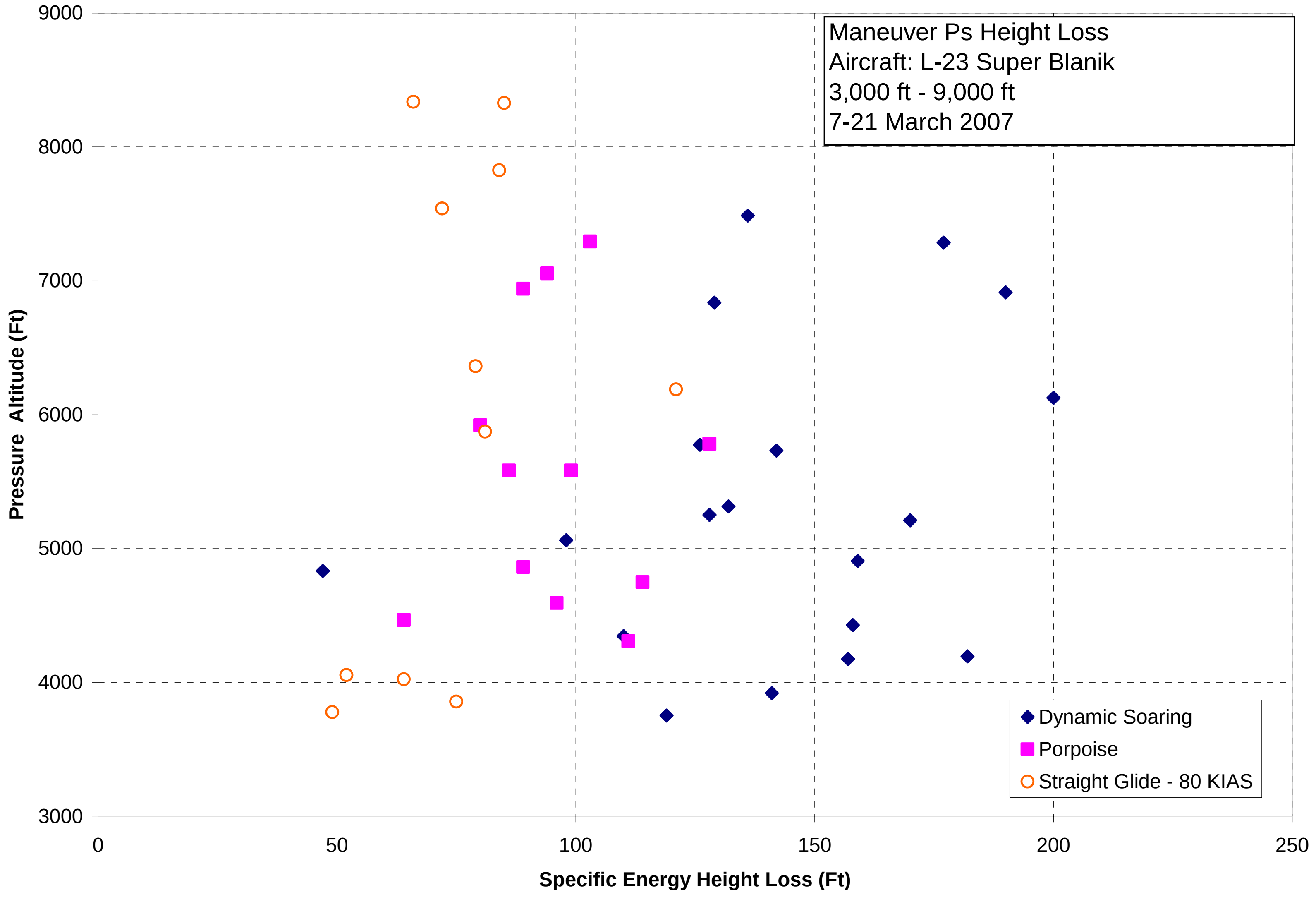


Figure D 3: Specific Energy Height Loss Raw Data

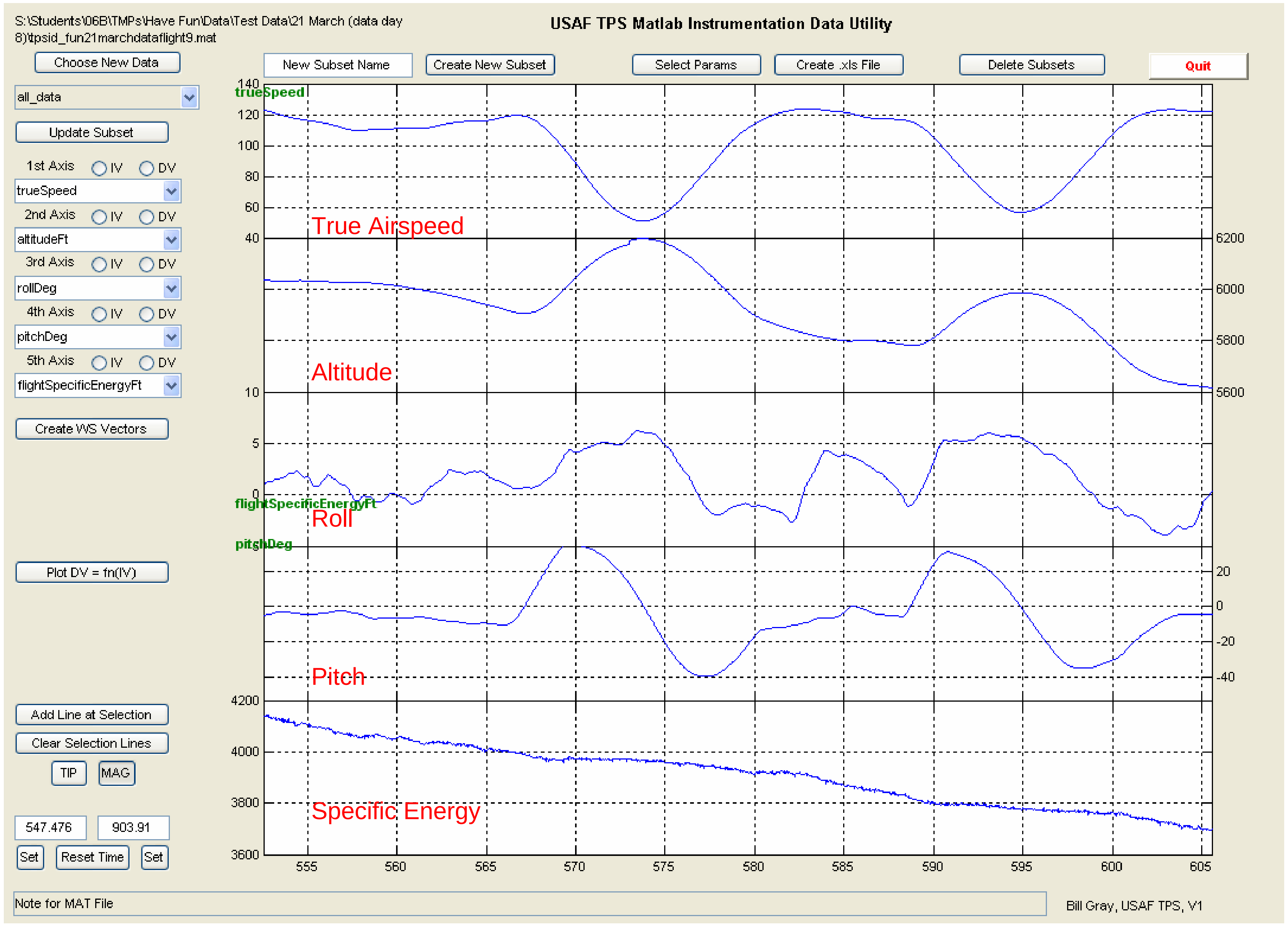


Figure D 4: Representative Porpoise Maneuver Raw Data

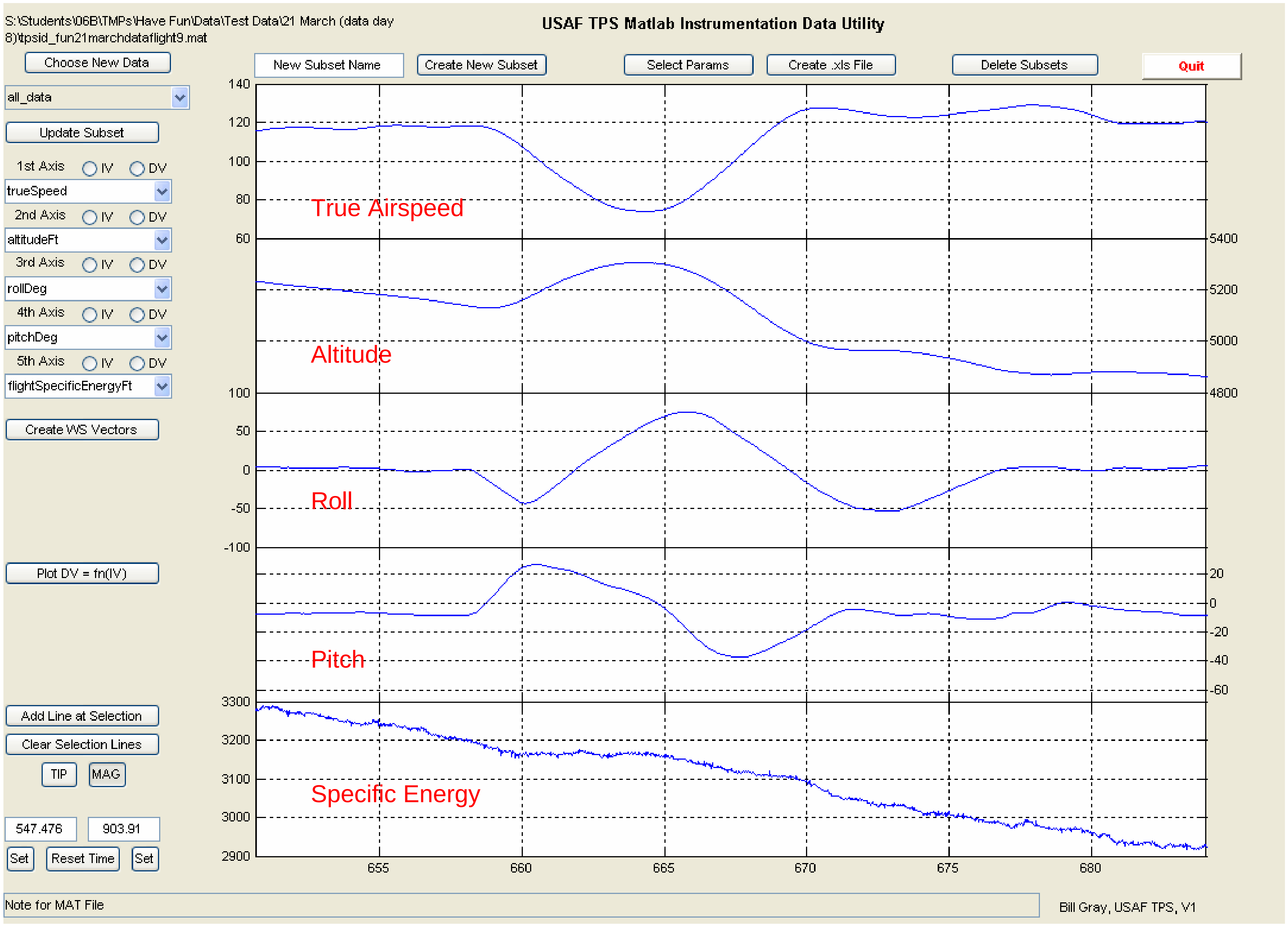

S:\Students\06B\TMPs\Have Fun\Data\Test Data\21 March (data day 8)\tpsid_fun21marchdataflight9.mat

Figure D 5: Representative Dynamic Soaring Maneuver Raw Data

Appendix E: Analytical Methods

The following is the method used to determine the estimated altitude lost during the nominal dynamic soaring maneuver duration due to the increased drag of control surface deflection. The only control surfaces accounted for in these calculations are the rudder and ailerons. The C_D of the elevator as well as the dynamic factors discussed in objective three are not considered in the following method.

Step 1:
The model from Senior IDS (reference 2) for C_L and C_D was used to determine values at an average α and average airspeed during the dynamic soaring maneuver.

$$\boxed{C_L = -.0029\alpha^2 + .0917\alpha + .6041}$$

For $C_L < 1.0$, $\boxed{C_D = 0.027{C_L}^2 + 0.017}$

Where: C_L = coefficient of lift
C_D = coefficient of drag
α = angle of attack = 0.5°

Step 2:
Determine aircraft drag using the following:

$$\boxed{D = C_D q S}$$

Where: D = drag
C_D = coefficient of drag
q = dynamic pressure
S = wing area = 206 ft^2

Step 3:
Determine the control surface contribution to drag for the average rudder deflection (10°) and average composite aileron deflection (9°) during the dynamic soaring maneuver:

$$\boxed{D_{surface} = q(S_{rudder} * C_{D_{rudder}} + S_{aileron} * C_{D_{aileron}})}$$

Where: D = drag
q = dynamic pressure
S_{rudder} = rudder surface area, 10.98 ft^2
$S_{aileron}$ = aileron surface area, 49.72 ft^2
$C_{Daileron}$ = coefficient of drag of the aileron (obtained from figure 5)
$C_{Drudder}$ = coefficient of drag of the rudder (obtained from figure 6)

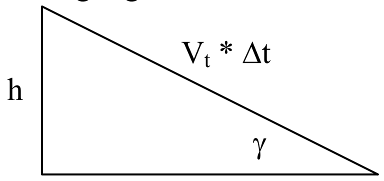
Step 4:
Calculate L/D using average load and weight of airplane + aircrew:

$$\boxed{\frac{L}{D} = \frac{N_{avg}W}{D}}$$

Where:	N_{avg} = average g load = 1.2 G
	W = average crew + airplane weight = 1069 lbs
	L = lift
	D = drag

Step 5:
Estimate the altitude loss over average dynamic soaring maneuver duration with additional aileron and rudder drag in a straight glide:
For gliders:

$$\boxed{\gamma = \tan^{-1}\left(\frac{-1}{L/D}\right)}$$



$$\boxed{\sin(\gamma) = \frac{h}{\mathrm{Vt} * \Delta t}}$$

Where:	L = Lift
	D = Drag
	h = altitude loss
	Vt = true airspeed
	Δt = duration of maneuver/glide

Appendix F: Modified Glider Aircrew Checklist

POWER ON

1. RADIO BATTERY – CONNECT; BATTERY CIRCUIT BREAKER – IN
2. DAS – CB IN AND SWITCH ON
3. ETHERNET/USB CABLES – CONNECT
4. PULL DAS SERIAL CABLE FROM DASHBOARD
5. FCP TABLET PC – ON
6. RCP TABLET PC – ON
7. AFTER FCP HAS WINDOWS DESKTOP – CONNECT DAS SERIAL CABLE AT DASHBOARD
8. FCP THUMP SOFTWARE RUN
9. FCP - IF NO TILTED ADI: DATA – INPUT – SERIAL – CONNECT
10. AFTER RCP HAS WINDOWS DESKTOP – FCP: DATA– OUTPUT– NETWORK– CONNECT
11. RCP THUMP SOFTWARE RUN, CLICK FLASH SCREEN, CLICK OK
12. RCP – DATA – INPUT – NETWORK - CONNECT
13. WIND DATA – INPUT

BEFORE TAKEOFF

1. ALTIMETER – FIELD ELEVATION
2. BELTS/HARNESSES – SECURE
3. CONTROLS/TRIM – CHECK (ENSURE FCP PC IS CLEAR OF CONTROLS)
4. CABLE – CHECK
5. CANOPIES – CLOSED AND LOCKED
6. DIVE/WHEEL BRAKES – CHECK
7. DIRECTION OF WIND – CHECK
8. EMERGENCIES – BRIEF
9. ES – SYNCHRONIZE
   RCP TOTAL ENERGY GAUGE – EMERGENCY ES ANNOTATE
10. TIME – RECORD

PRE TEST MANEUVER

1. BRIEF – AIRSPEED, PULL/PUSH G TARGETS/LIMITS
2. BELT/HARNESSES – SECURE
3. DATA - ON

POST TEST MANEUVER

1. DEBRIEF – MVR QUALITY / EVAL COMMENTS
2. DATA - OFF

BEFORE LANDING

1. TRAFFIC
2. WINDS
3. RUNWAY
4. SPEED
5. SPOILERS

POST-FLIGHT

1. TIME – RECORD
2. TURN OFF RADIO AND TRANSPONDER
3. RCP – THUMP SOFTWARE: FILE – EXIT…….. SHUTDOWN TABLET PC
4. FCP – THUMP SOFTWARE: FILE – EXIT…….... DOWNLOAD DATA TO THUMBDRIVE
5. FCP – MOVE DATA FROM DATA FOLDER TO RECYCLE BIN…….. SHUTDOWN TABLET PC
6. PULL SERIAL CABLE FROM DASHBOARD
7. SWAP TABLET PC BATTERIES
8. IF FLYING AGAIN: GO TO POWER ON CHECKLIST STEP 5

END OF FLYING

1. ATHENA POWER/GPS CB – OFF
2. ETHERNET/USB CABLES – DISCONNECT
3. TABLET PCS – REMOVE
4. RADIO BATTERY – REMOVE
5. DAS BATTERY – REMOVE
6. ENSURE TOWPLANE FLIGHT LOG FILLED IN

Appendix G: Acronym and Abbreviation List

Abbreviation	Definition
AFB	Air Force Base
AFFTC	Air Force Flight Test Center
AFMC	Air Force Material Command
AGL	above ground level
AOA	angle of attack
AOSS	angle of sideslip
Avg	average
C	Centigrade or Celsius
CA	California
Capt	Captain
CB	circuit breaker
CD	compact disk
C_L	coefficient of lift
C_D	coefficient of drag
$C_{Daileron}$	coefficient of drag due to aileron deflection
$C_{Drudder}$	coefficient of drag due to rudder deflection
COOL	center operations online
DAS	Data Acquisition System
DFRC	Dryden Flight Research Center
DO	Director of Operations
DOE	design of experiments
Deg	degree(s)
E_s	energy height
FCP	front cockpit
Ft	feet
FUN	FUll Scale Numbers
g	acceleration due to gravity
GPS	global positioning system
GS	Guidestar
GUI	graphical user interface
HOS	hands on stick
Hz	hertz
IMU	inertial measurement unit
INS	inertial navigation system
IP	initial point; instructor pilot
In	inch(es)
JON	job order number
KCAS	knots calibrated airspeed
KIAS	knots indicated airspeed
kt	knot(s)
lb	pound(s)
L/D	lift to drag ratio
LED	light emitting diode
Maj	Major
Max	maximum

Abbreviation	Definition
MHz	megahertz
MSL	mean sea level; missile
NASA	National Aeronautics and Space Administration
NCO	non-commissioned officer
No.	number
NOTAM	notice to airmen
OPS	operations
ORI	operational readiness inspection
PC	personal computer
RC	remote control
RCP	rear cockpit
RTD	resistive temperature detector
Sec	second(s)
TACAN	tactical aid to navigation
TAS	true airspeed kt
TIM	technical information memorandum
TMP	test management project
TPS	Test Pilot School
USA	Untied States of America
USAF	United States Air Force
USN	United States Navy
VA	Virginia
VHF	very high frequency

Appendix H: Lessons Learned

Tablet PC

The startup procedures for the DAS and Tablet PC software are not concrete since multiple variations were used with success. However, there were times where we lost more than an hour trying to get the system online with the computers functioning properly. Probably the biggest gotcha is trying to get the tablet PCs to start the ThUMP software before the entire boot up sequence was finished after initial power up (i.e. the blue light indicating hard disk operation was still flickering when the software was being started). Users must be patient and allow the startup sequence to complete before attempting software initialization.

If you receive a “Error – Could not open COM3” on the front computer, turn it off, wait a few minutes and try the process again. Changing USB ports or reinitializing the software without a cold reboot was fruitless.

If you receive a “Error – Could not open COM1” on the rear computer during software initialization, acknowledge and continue. This error was a nuisance.

Before testing, ensure all channels of the DAS are operating correctly by looking at the “Quicklook” screen and verifying each data stream during a flight control sweep. Do a max deflection control sweep as the first maneuver in your test flight if you need deflection data for your TMP. The zero deflection and max deflection values found during the ground block did not match with data taken in the air. This makes it extremely difficult to ensure you have proper DAS calibration.

Change the tablet PC batteries out every flight. If the batteries fail in flight, the DAS data file produced will contain corrupted data.

Some of the DAS data is very noisy. In particular, the true airspeed, angle of attack, angle of sideslip, and outside air temperatures should be filtered heavily before use in data reduction.

Avionics

During flight test, the first indication of impending battery depletion was a flickering ‘TX’ light on the radio control head. The transmitter sounded like multiple “zippers” were being commanded. The aircraft battery still indicated 3 red LEDs when the test button was pressed. During test, it was determined that anytime less than 4 LEDs were present during a battery test, the radio would malfunction in the described manner. Always have a charged backup battery.

Scheduling

Forecasting conflicts with your test schedule should include the following: the ED calendar, squadron scheduling, the secretary’s calendar for squadron events, AFMC family days, base exercises and ORI cycle, Qual Eval schedule, no-fly Tuesdays at Tehachapi, and tow pilot availability. Do not expect a full team at each scheduled soaring period during your testing window. Flexibility is the key to success and keeping the scheduling shop in loop (daily) is required.

The Director of Operations (DO) did not allow flying operations without a staff instructor present at the lakebed. If the tow pilot was a staff instructor (Mr. Gary Aldrich or Mr. Roger Tanner), the tow pilot filled the DO's requirement. If the tow pilot chosen was not an instructor, the staff glider instructor would be underutilized while at the lakebed. To reduce wasted manpower, attempt to schedule a staff tow pilot.

Scheduling of the glider TMP operations is non-standard. When the schedule is created and posted in COOL, ensure that the pilot who will be signing out the line at the OPS desk has the "A" code. When flying is complete, anyone can fill out the 781 and sign the group back in. Be sure to keep a record of the takeoff and land times for each sortie to ease the sign in process. Since these are test sorties (O-4, not T-1A) and count toward aerial achievement medals, ensure the proper number of sorties for each student are logged accurately to prevent a sortie accounting headache at the end of the project.

Every aspect of scheduling the tow plane should include both Browne's Aviation and Mrs. Jane Barrett at the Tehachapi soaring operation. While Mr. Bob Browne owns the airplane, Mrs. Jane Barrett plans her commercial activities based on tow plane availability.

Keep a meticulous log of crews and associated flight times. There is no accounting structure in place for glider operations other than your test team.

Logistics

The Aero Club lent us a fuel truck for the entire testing window. The club had to be reminded of the agreed delivery date after testing had already begun – be sure to remind all players necessary to begin operations about 2 days in advance. They may be understaffed and need help getting the truck to North Base. Ensure a few people receive training on how to operate the fuel truck in case the tow pilot is unfamiliar. If TPS had Qual Eval aircraft that required use of the truck, they will receive priority. If the fuel truck was not available, the tow plane will have to fly to South Base to refuel. The Aero Club usually provides a clipboard and log in the cab of the fuel truck to record fuel consumption. Ensure the tow pilot is updating the log.

Before the sailplane launches on its return trip back to Tehachapi, make sure the cockpit canvas cover, control locks, and pitot covers are in the aircraft.

There are tarps available at TPS to cover the sailplane and tow plane while being stored at North Base. Also, a government vehicle is usually available for logistical support during testing from the Vehicle NCO. Ensure that all team members are aware of normal government vehicle use to include signing the preflight form each day. Each team member needs to get a flightline driver's license to simplify vehicle logistics.

Get a handheld radio a few days prior to testing to prevent other TMP groups from monopolizing all of the TPS resources. Each group will need at least one handheld VHF radio for contact with ground control during movement on the flightline. We ended up using a personal handheld radio to conduct communications with the tower.

Closing North Base Operations

When testing is complete and the sailplane is no longer in R2508, complete the following:

1. Call Base Operations and have the glider NOTAM closed
2. Inform the Director of Operations
3. Gather the fuel log and make a copy before coordinating the fuel truck's return to the Aero Club (if used).
4. Clean out the hanger and leave it the way you found it
5. Inform scheduling that glider flying at North base in complete
6. Return all of the tablet PCs, aircraft batteries, and chargers to there proper storage locations.

This page intentionally left blank.

For onsite distribution:	Paper / CD
812 TSS/ENTL 307 E Popson Ave, Bldg 1400, Rm 110 Edwards AFB CA 93524-6630	3 / 1
USAF TPS/EDT Attn: Mr Gary Aldrich 220 S Wolfe Ave, Bldg 1220 Edwards AFB CA 93524-6485	2 / 1
USAF TPS/DOQ Attn: Maj Chad E. Ryther 220 S Wolfe Ave, Bldg 1220 Edwards AFB CA 93524-6485	1 / 1
USAF TPS/EDC Attn: Ms Dottie Meyer 220 S Wolfe Ave, Bldg 1220 Edwards AFB CA 93524-6485	2 / 1
773 TS/ENFB Attn: Mr. Fred Webster Building 1400, Room 102 307 East Popson Avenue Edwards AFB CA 93524	1 / 1

For offsite distribution:	Paper / CD
Defense Technical Information Center 8725 John J. Kingman Rd, Ste 0944 ATTN: Willis Smith (DTIC-OCA) Fort Belvoir, VA 22060-6218	1 / 1
AFIT Academic Library ATTN: Librarian Bldg 642 2950 Hobson Way Wright Patterson AFB, OH 45433-7765	1 / 1
Mrs. Jane Barrett P.O. Box 100 Tehachapi, CA 93581	1 / 1
Capt Brent Reinhardt 6704 Doolittle Dr Edwards, CA 93523	1 / 1
Maj Jeff Geraghty 6841 Lindbergh Ave. Edwards, CA 93523	1 / 1
Maj Sean Celi 414 12th St. Edwards, CA 93523	1 / 1
Maj James Stahl 6747 Rickenbacker Dr. Edwards, CA 93523	1 / 1
LT Victor Glover 6779 Rickenbacker Dr. Edwards, CA 93523	1 / 1
Capt Geoff Bowman 425 11th St. Edwards, CA 93523	1 / 1
James Murray NASA Dryden Flight Research Center PO Box 273 MS 4840 Edwards, CA 93523-0273	1 / 1
Russ Franz NASA Dryden Flight Research Center PO Box 273 MS 2306 Edwards, CA 93523-0273	1 / 1